A User's Guide to Estimating Total Mortality Rates from Length Frequency Data with the Method of Fournier and Breen

Paul A. Breen and David A. Fournier

Department of Fisheries and Oceans
Fisheries Research Branch
Pacific Biological Station
Nanaimo, British Columbia V9R 5K6

January 1984

Canadian Technical Report of Fisheries and Aquatic Sciences No. 1239

Canadian Technical Report of Fisheries and Aquatic Sciences

These reports contain scientific and technical information that represents an important contribution to existing knowledge but which for some reason may not be appropriate for primary scientific (i.e. *Journal*) publication. Technical Reports are directed primarily towards a worldwide audience and have an international distribution. No restriction is placed on subject matter and the series reflects the broad interests and policies of the Department of Fisheries and Oceans, namely, fisheries management, technology and development, ocean sciences, and aquatic environments relevant to Canada.

Technical Reports may be cited as full publications. The correct citation appears above the abstract of each report. Each report will be abstracted in *Aquatic Sciences and Fisheries Abstracts* and will be indexed annually in the Department's index to scientific and technical publications.

Numbers 1-456 in this series were issued as Technical Reports of the Fisheries Research Board of Canada. Numbers 457-714 were issued as Department of the Environment, Fisheries and Marine Service, Research and Development Directorate Technical Reports. Numbers 715-924 were issued as Department of Fisheries and the Environment, Fisheries and Marine Service Technical Reports. The current series name was changed with report number 925.

Details on the availability of Technical Reports in hard copy may be obtained from the issuing establishment indicated on the front cover.

Rapport technique canadien des sciences halieutiques et aquatiques

Ces rapports contiennent des renseignements scientifiques et techniques qui constituent une contribution importante aux connaissances actuelles mais qui, pour une raison ou pour une autre, ne semblent pas appropriés pour la publication dans un journal scientifique. Il n'y a aucune restriction quant au sujet, de fait, la série reflète la vaste gamme des intérêts et des politiques du Ministère des Pêches et des Océans, notamment gestion des pêches, techniques et développement, sciences océaniques et environnements aquatiques, au Canada.

Les Rapports techniques peuvent être considérés comme des publications complètes. Le titre exact paraîtra au haut du résumé de chaque rapport, qui sera publié dans la revue *Aquatic Sciences and Fisheries Abstracts* et qui figurera dans l'index annuel des publications scientifiques et techniques du Ministère.

Les numéros 1-456 de cette série ont été publiés à titre de Rapports techniques de l'Office des recherches sur les pêcheries du Canada. Les numéros 457-714, à titre de Rapports techniques de la Direction générale de la recherche et du développement, Service des pêches et de la mer, ministère de l'Environnement. Les numéros 715-924 ont été publiés à titre de Rapports techniques du Service des pêches et de la mer, Ministère des Pêches et de l'Environnement. Le nom de la série a été modifié à partir du numéro 925.

La page couverture porte le nom de l'établissement auteur où l'on peut se procurer les rapports sous couverture cartonnée.

84- 2872 /

Canadian Technical Report of

Fisheries and Aquatic Sciences No. 1239

January 1984

A USER'S GUIDE TO ESTIMATING TOTAL MORTALITY RATES FROM LENGTH FREQUENCY DATA WITH THE METHOD OF FOURNIER & BREEN

by

Paul A. Breen

Department of Fisheries and Oceans

Fisheries Research Branch

Pacific Biological Station

Nanaimo, British Columbia V9R 5K6

and

David A. Fournier

Statistics Canada

Structural Analysis Division

Tunney's Pasture, Ottawa, Ontario

Canada K1A 0T6

(c)Minister of Supply and Services Canada 1984

Cat. No. Fs 97-6/1239 ISSN 0706-6457

ABSTRACT

Breen, Paul A. and David A. Fournier. 1984. A user's guide to estimating total mortality rates from length frequency data with the method of Fournier & Breen. Can. Tech. Rep. Fish. Aquat. Sci. 1239: iii + 63 p.

This report is a user's guide to the method of Fournier & Breen (1983) for simultaneous estimation of growth and mortality rates from population length frequencies. The program is described, required input values are discussed in detail, and an example is worked through. This example uses data produced by a population length frequency simulator, which is also described. Some general guidelines are given to guide the reader toward a strategy of finding realistic solutions, and some basic troubleshooting is offered. The source code for both the program and the model is given in appendices.

Key words: growth, mortality, non-linear parameter estimation

RÉSUMÉ

Breen, Paul A. and David A. Fournier. 1984. A user's guide to estimating total mortality rates from length frequency data with the method of Fournier & Breen. Can. Tech. Rep. Fish. Aquat. Sci. 1239: iii + 63 p.

Le présent rapport constitue un guide d'utilisation de la méthode de Fournier et Breen (1983) visant l'estimation simultanée des taux de croissance et de mortalité à partir de fréquences de longueur d'une population. Les auteurs décrivent le programme, expliquent les entrées nécessaires et appliquent la méthode à un exemple que utilise des données générées par un simulateur de fréquences de longueur d'une population, lequel est aussi décrit. Ils donnent des directives générales afin de guider l'utilisateur vers une stratégie permettant la découverte de solutions réalistes et ils fournissent certains moyens fondamentaux pour la détection des problèmes. Le code de source du programme et du modèle est inclus en annexe.

Mots-clés: croissance, mortalité, estimation de paramètres non linéaires.

1. INTRODUCTION

This guide is aimed at the biologist who wants to estimate mortality rate and growth parameters from length frequency data. The model is that of Fournier & Breen (1983). The user should become familiar with the model as described there in detail before this guide is used.

The model was developed as an extension of the general approach outlined by Schnute & Fournier (1980). The size structure of a population is a product of several dynamic processes: annual recruitment, mortality, growth, and variability in those processes. Whereas the traditional approach has been to estimate one parameter from information already obtained about the others, the new technique described here is to simultaneously estimate all the parameters of interest.

The method is especially useful where ageing is not possible. It was developed to solve the problem of estimating total mortality rate in northern abalone (Haliotis kamtschatkana), which cannot be aged. The model is applicable to any species whose biology conforms with the assumptions.

The model itself consists of a set of assumptions about the processes which shape the length structure of a population (see Fournier & Breen 1983). The model assumes that growth follows the von Bertalanffy model:

$$\ell_i = L_\infty(1-e^{-K(i-t_0)}) \qquad (1)$$

where ℓ_i is length at age i, L_∞ is asymptotic length, t_0 is the age at which a fish would have been zero length if growth were von Bertalanffy in form from zero size, and K is the Brody coefficient (the instantaneous rate of approach to L infinity). The model assumes that the distribution of lengths around the mean length at age is normal, and that the standard deviations of length at age are related to age according to:

$$s.d._i = a + b\sqrt{i} \qquad (2)$$

where $s.d._i$ is the standard deviation of length at age i and a and b are constants. The model assumes that the abundance of individuals decreases over time according to:

$$N_i = N_{i-1}\, e^{-Z} \qquad (3)$$

where N_i is relative abundance at age i and Z is total instantaneous mortality rate. Thus the model fits three situations: unfished populations (Z=M), populations fished for a long time at a constant rate, and virgin populations fished for only a very short time.

The program uses a non-linear parameter estimation procedure to fit a set of parameter values to the set of equations which make up the model. The program obtains the best fit of such values to the actual data, which are length frequency observations. In doing so, the program makes simultaneous estimates of growth, mortality and variability within the population.

The input to the program is a set of observed (or simulated) length frequencies; i.e. the number of individuals observed within each of a number of length intervals. The program attempts to find a set of parameter values which are most consistent with the data. These values include: a set of mean lengths at age; a set of standard deviations for each mean length; a set of population proportions (i.e. the proportion of the population contained within each age class); growth parameters: L infinity, Brody coefficient K, and t zero of the von Bertalanffy growth equation; and a total mortality rate Z. (If the user is unfamiliar with these concepts, he should read the appropriate sections of Ricker (1975)).

The program finds its solution by calculating an objective function (one can think of this as measuring the goodness of fit between the solution and the input data) and varying the parameters involved in order to maximize the goodness of fit. The concept of an objective function is discussed by Schnute and Fournier (1980). This function is based on the log-likelihood function, which is the basis of the method (see Fournier and Breen 1983). For a given set of input and input options discussed below, the goodness of fit is best when the objective function is minimized.

There are three kinds of input to the program: the length frequencies, a set of initial values for some of the parameters (from these the program calculates other initial parameter values), and bounds beyond which parameter values are not permitted to go. Using initial values, the program calculates parameters that can be changed to decrease the objective function at the highest rate, and the direction of such changes. If one were trying to find the minimum value on a curve in a simple two-dimensional situation, the partial derivatives are related to the slope of the curve, and are zero at places where the curve 'bottoms out'.

In the beginning and at subsequent intervals, the program calculates the objective function using the current parameter values: this is called a function evaluation. In between function evaluations, the program performs a procedure to determine which should be varied in which direction to reduce the function. Then it tentatively changes the values, does a function evaluation, and checks to determine whether the function actually decreased. If it did, the new parameter values are used as the basis for the next step, and the process from the last similar step is termed an iteration. If the function does not decrease, a different set of tentative parameter value changes is made, then the function is evaluated, and so on. Each iteration must constitute at least one function evaluation (and need include no more if the solution is being approached), but in more difficult cases may constitute many function evaluations.

In this program, the process proceeds in several phases, and a provisional solution is printed at the end of each phase. In the first phase, only the population proportions are allowed to vary; in the next phase, the mean lengths are allowed to vary as well. In the third phase, the constant a in equation (2) above is allowed to vary along with the population proportions and mean lengths. During the last phase, the program is free to vary any parameter, and the solution printed at the end of this phase will have the lowest objective function value. Phasing forces the program to make estimates first of those parameters that are least well known, while using the user's initial estimates for parameters that are better known.

It will become clear that the program can provide a large set of different (and possibly conflicting) solutions when the input constraints are varied. The user should keep in mind that while the goal of the program is to minimize an objective function, his own goal is different - to find a realistic solution that conforms well with those aspects of the species' biology that are known.

In what follows, we have tried to anticipate the needs of both users of the system at the Pacific Biological Station (to whom we refer below as 'PBS VAX USERS') and users on other systems. Comments which are specific to the PBS Vax machine are identified as such; users of other systems may find them useful by analogy or can ignore them.

The source for the program is presented as Appendix 1; the source code for the population length frequency simulator is presented as Appendix 2. If the reader sends a tape to P. A. Breen at Nanaimo, it will be returned unlabelled with the source code in either EBCDIC or ASCII, and either 800 or 1600 bpi (please specify). Users on other systems may find that small amounts of programming changes are required to make the code compatible with their system. The code is not as neat as one would like: it is the product of a long period of modification and revision. Ideally, someone should have taken the final model and programmed it from scratch; since that isn't going to happen we present the code we have.

2. INPUT TO THE PROGRAM

Table 1 shows a file of input to the program. At the head of the source code (Appendix 1) is a brief table giving the names of values as used in the program, their variable type, and meaning. Each variable is discussed in greater detail below.

The program reads all input values in free-format. The number of values must be internally consistent. If the number of age classes is set at 12, the program will read 12 values for the lower and upper bounds on the standard deviations, the population proportions, and the mean lengths. It will read the number of length frequencies specified. If the number of values offered is too small or too large, the program will be reading as input different values from what was intended.

FIRST RECORD:

NK (The number of age classes).
The program can handle up to 16 age classes inclusive. If you knew for sure how many age classes were present, perhaps from ageing information, you

wouldn't be using this method. As the program doesn't vary this parameter, you'll want to make several runs with different values of NK. If you have too few age classes, the program cannot produce a 'good' solution - one that conforms well with the data.

MAXFN (The maximum number of function evaluations).
This determines the maximum number of function evaluations before the run terminates. Its function is to prevent the model from getting stuck and hanging up indefinitely. A good value, one which usually does not affect the normal termination, is 400.

NFULL (The number of the first fully recruited age class).
The program is going to estimate growth parameters and total mortality simultaneously. In many fishery and population studies the youngest, smallest individuals are not caught by the fishery or sampled by the biologist with the same efficiency as larger, older individuals. This doesn't affect the estimation of growth parameters, but it does affect the estimation of total mortality. This parameter allows the user to exclude the first few age classes from the procedure which estimates total mortality.

Here is an example. If the first lengths in the input are from age three individuals, then the first age class in the model represents age three, the second age four, and so on. If the first age class which you consider fully recruited to your sampling procedure is age five, that is the third age class in the model, so the value for NFULL should be 3. NFULL represents the number of the first fully recruited age class, when the first age class in the model is called 1.

NFULL must obviously be less than NK, and as it approaches NK the total mortality estimates become meaningless.

SECOND RECORD:

FMORT determines whether total mortality is estimated, or whether only growth parameters are estimated. Setting FMORT to a positive value less than 1.0 cause total mortality to be estimated; setting FMORT to 0 causes it not to be estimated.

PENMOR (The penalty associated with total mortality rate estimates). The program is trying to fit the input data both to a growth model and to a mortality model, and the noise in the data make it difficult for the program to do both things. This penalty, and the one associated with the distribution of mean lengths along the ideal von Bertalanffy curve, allow the user to guide the program through the various possible alternatives:

1) with a high penalty PENMOR and a low penalty PP, the program will try hard to arrange the population proportions along an exponentially decaying sequence, at the expense of the spacing of the estimated mean lengths at age.

2) with a high penalty PP and low penalty PENMOR, the converse occurs: the program arranges the mean lengths close to a von Bertalanffy curve, and the sequence of population proportions probably varies from ideal exponential decay.

3) if both penalties are very low, the program is free to arrange the mean lengths and the population proportions in whatever way minimizes the function; even if the solution doesn't bear much resemblance to the notions of von Bertalanffy growth and exponential decay. This situation yields the lowest objective functions but probably the least realistic solutions.

4) when both penalties are high, the program tries equally hard to satisfy both requirements. Objective functions will be high.

The user should experiment with these opposing penalties to observe their effect. A larger value of PENMOR implies a larger penalty.

THIRD RECORD:

IPRINT (Maximum number of iterations per printout).
If IPRINT is 0, then no statistical information is published by the function minimizer. If IPRINT is positive, then the statistics are published on unit 109, either at the end of each phase or at iteration IPRINT, whichever comes sooner. The ordinary user will probably have no use for the statistics, and so will set this to 0.

FOURTH RECORD:

N5 (The number of length frequency observations).
If the observations are arranged 10 to a line, then they are easy to count when one checks that N5 agrees with the data. The maximum permitted is 85.

W5 (Age of the first age class). This affects only the estimate of t zero.

S5 (The lower edge of the first length interval).
S6 (The width of each length interval).
Taken together, these tell what the length frequency observations mean. If S5 were 1 and S6 were 2, the first observation would include individuals with lengths from 1.0 to 2.999, the second includes individuals from 3.0 to 4.999, and so on. Length intervals should be as small as practical - the program will have difficulties when the interval width approaches the estimated standard deviation of lengths at age. In creating files of length frequency observations, some users may want to start always at length 0.0, even though the first observation may be at some greater length. Outside the program, this makes comparing different data and plotting the data a little easier.

FIFTH AND SUBSEQUENT SEVERAL RECORDS:

FREQ (Frequency observations). Each value is the number of individuals observed within a particular length interval, defined by the values given in the fourth record. Normally one observes whole numbers of individuals, but the program accepts decimal individuals, such as might be produced by stratified sampling.

NEXT RECORD:

FKMIN (The lower bound on the Brody coefficient).
As discussed in Fournier & Breen (1983), the model has some tendency to under-estimate the Brody coefficient K. FKMIN can be set as the lowest value of K acceptable to the user. Since the in the final solution the program often estimates K equal to FKMIN (this is termed 'being tight on the lower bound' or 'on the peg') the user may set FKMIN to a particular desired value of K.

PP (The penalty associated with varying the mean lengths). See the discussion above for PENMOR. A larger value implies a larger penalty.

SDMIN, V1MAX, VKMIN, VKMAX (Bounds on standard deviations of lengths at age). If the standard deviations were not bounded, the program might make estimated standard deviations unrealistically large in order to obtain a good fit. Standard deviations are usually known or can be estimated from inspection of the data. Initially, these bounds should set quite wide; if the estimated standard deviations become too large the bounds can then be used to force a more credible solution.

NEXT TWO RECORDS:

PMIN, PMAX (lower and upper bounds on population proportions). These should be set very wide - 0.0 and 1.0 respectively - for the initial runs, and can usually be left there. In some cases, the user may find that the program persistently under- or over-estimates the proportion of one age class. In that case, the bounds for that age class can be used to force a more realistic solution.

NEXT TWO RECORDS:

FMMIN and FMMAX (The lower and upper bounds on the mean lengths). These bounds can also be set initially, as Table 1 shows, to have no effect on the estimated mean lengths. In particular cases, the program may find a solution

having an estimated mean length with which the user disagrees. This will probably be one of the early age classes, or the last age class. These bounds can be used to force a realistic solution.

NEXT RECORD:

X(1), X(2), X(3), X(4), X(5)
X(1) and X(2) (The initial value for mean lengths of the first and last age classes). The user estimates these from his data; then the program uses X(3) (the initial value of the Brody coefficient) to calculate the initial mean lengths of all the intermediate age classes. Similarly, the program takes X(4) and X(5) as the initial values for the standard deviations of the first and last age classes, and calculates the initial values for the other age classes.

These values can be thought of as giving the program a place to start: they are used in finding the solution which is published at the end of the first phase, but they should not be critical to the final solution.

Each of these estimates must, however, be consistent with any bounds that have been set. X(3) must be larger than FKMIN; X(4) must fall within the bounds set by the first values of SDMIN and V1MAX, etc.

NEXT RECORD:

X(J) (Initial values for population proportions).
Like the values above, these give the program a place to start, and no precise estimates are required. The estimates must be consistent with the bounds set by PMIN and PMAX (a good reason to make the bounds wide); and they also must sum to less than 1.0.

The user must supply NK-1 initial values for population proportions, because the program finds the last value by subtraction. If there are 11 age classes, only 10 values are required.

LAST RECORD:

TITLE. This is just a record that identifies the data, and perhaps says something about the input. The user has 80 bytes to express himself with complete freedom.

3. PREPARING THE INPUT FILE

The user has a set of length frequencies either from population sampling or from a simulator (Table 9). In working through this example, we have used the output from a population length frequency simulator described further below. Since one knows the 'true' parameters of a simulated population, the use of simulated data is a valuable guide to the performance of the model. In making the required initial estimates, it is very useful to have a length frequency plot (Figure 1).

First, the user chooses NK, the number of age classes, based either on the data or on some other knowledge of the biology. In this example, we will use 10. Set MAXFN at 400. Next, which is the first fully recruited age class? On the plot, it appears that the first age-class is distinct, and it appears that its size greater than the next age class. But perhaps the biologist still suspects sampling difficulties - for the example we'll use a value of 2.

Setting FMORT to 0.5 causes total mortality rate to be estimated, and it is best for the first run to use a fairly low penalty such as 1.0. The penalties can be varied later if necessary to force more realistic solutions.

IPRINT should initially be set to 30 unless the user has absolutely no interest in the stat report, in which case he can set it to 0.

The next line contains the number of length frequencies, the first age, the lower edge of the first length interval, and the width of length intervals. Following these are the length frequencies, in this case produced by the simulator described further below.

We must next find an estimate for the Brody coefficient, unless one is available from some other work. Figure 2 shows a crude but effective method. The first two peaks, which might represent the first two age classes, seem to be near 32 and 56. The maximum size observed was about 125; this might be taken as L infinity. A Ford-Walford plot (Ricker 1975) can be constructed from these three points as Figure 2 shows - not one that anybody would take to a meeting, but one good enough to give the program a starting value. The slope (rise over run) is 0.74. This is the Ford coefficient, and the Brody coefficient is given by:

$$K = -\ln k$$

which yields a value of 0.30. Let's therefore start with a lower bound of 0.20. The program usually under-estimates the Brody coefficient, so we can work up on it. For a penalty, start with a low value of 1.

For the bounds on the standard deviations, look at the plot. The first age class seems to extend from roughly 22 to 40, a range of 18. Remembering that 2 standard deviations each side of the mean include 95% of a normal distribution, we can estimate the standard deviation of the first age class to be 4.5. So let's first set the bounds wide - to 1.5 and 7.5 and then see what happens. The standard deviation of the last age class might be larger, so we can use 2.0 and 10.0 in this first run.

For the bounds on proportions and mean lengths, one might as well make bounds so wide that they have no practical effect, especially for the first run. We'll use 0.0 and 1.0 for proportions; 0.0 and 200.0 units for mean lengths.

We next have to estimate the first and last mean lengths - the first peak is at 32 units, and the last observations are centred at 120, so we'll use those as starting values. The exact values used here should have no impact on the final solution. The Brody coefficient was estimated above to be 0.30, and the first standard deviation to be 4.5. There is no way to estimate the last standard deviation from the data; it should be somewhat larger than the first. Say 5.5, so the difference between them is 1.0.

Finally, we estimate the population proportions. Again, these values should have no impact on the final outcome, although better values might give a more efficient run. The important consideration is to have nine values (because there are 10 age classes) that add up to less than 1.0. The values we used are shown in the input file (Table 1). The last record is the title.

4. RUNNING THE PROGRAM

The program reads input from unit 105, writes the main output to unit 108, and writes some statistics to unit 109.

PBS VAX USERS: Assign input and output files as follows:

```
$ASSIGN INVB.DAT  FOR105
$ASSIGN OUTVB.LIS FOR108
$ASSIGN STATS.DAT FOR109
$RUN USERLIB:VBM
```

On the PBS Vax, the user will find that running the program remotely through the batch queue is more efficient than running interactively. Here is a sample command file:

```
$! SET DEFAULT TO YOUR DIRECTORY:
$SET DEF [TROLL]
$! ASSIGN THE INPUT FILE NAME TO 105:
$ASSIGN INVB.DAT  FOR105
$! ASSIGN THE OUTPUT FILE NAMES TO 108 AND 109:
$ASSIGN OUTVB.LIS FOR108
$ASSIGN STATS.DAT FOR109
$! RUN THE PROGRAM:
$RUN USERLIB:VBM
```

OTHER USERS:

You'll have to figure out how, on your own system, to get the input file read by unit 105, and how to handle the two output files on units 108 and 109. It is even possible that your system won't handle unit numbers as large as this, in which case you'll have to change the read and write statements in the program.

ALL USERS:

The main output file is on unit 108. The first page reports the input values and calculated initial values (Table 2). The user should always check this page carefully to ensure that the input file was correctly read. At the top of the page the program prints the date and time, so that this file can be matched up with the stats file. The next point to note is that the program calculated the mean lengths from the initial values for the first and last, and similarly calculated the standard deviations from initial values. The program added up the length frequencies to get a total.

The second output page (Table 3) is headed by the title record. The solution on this page was obtained at the end of the first phase, in which only the population proportions were allowed to vary. The estimated mean lengths and standard deviations are therefore the same as on the first page. Because the means were held fixed, the estimated Brody coefficient is the same as the initial value.

In the second phase, the mean lengths are allowed to vary but standard deviations are still held. The solution (Table 4) shows different estimates for the von Bertalanffy parameters, and has slightly changed the estimated population proportions. These changes have produced a big decrease in the objective function.

In the third phase, the parameter a of equation (2) determining standard deviations of lengths at age is permitted to vary (Table 5).

By the end of the last phase, with all parameters varying, the program has managed to find progressively better solutions (Table 6.). The following tables summarize this solution with the 'true' values (which were specified as input to the data simulator).

	L_∞	K	t_0	a	b	Z
'true' value	130.0	0.25	0.00	5.00	0.20	0.245
model estimate	130.1	0.26	-0.08	5.17	0.16	0.197

age	1	2	3	4	5	6	7	8	9	10
'true' mean length	31.3	54.9	73.9	88.1	99.8	108.8	115.9	120.9	-	-
model estimate	31.8	54.6	71.8	84.8	95.6	103.3	109.5	114.3	117.9	120.6

age	1	2	3	4	5	6	7	8	9	10
population %s	0.38	0.16	0.14	0.12	0.09	0.10	0.06	0.04		
model estimate	0.35	0.15	0.12	0.09	0.07	0.08	0.04	0.04	0.04	0.03

In the final solution, the program has estimated the von Bertalanffy parameters quite well, and has come close to the true standard deviations. It has estimated the early population proportions and mean lengths well; but because it is trying to make too many age classes, it does poorly with the later age classes. Total mortality rate is somewhat underestimated, and indicates 83% annual survival rather than 78%.

This solution seems reasonably close to the values we originally put into the simulator, even though we purposely used the wrong number of age classes (10 instead of 8), left the bound on the Brody coefficient low (0.20 instead of 0.25), and made NFULL 2 when it could as well have been 1. In particular, the program has arrived at its solution without being constrained by any of the bounds.

The statistical report is presented here only for the user who wishes to use it for troubleshooting; all other users will probably be content to turn it off by setting IPRINT to 0, and they may ignore this discussion as well. The first page of the statistical report is shown in Table 7. The first part of the report was printed at the beginning of the first iteration. At the very top are the time and date, which correspond with those on the main output (Table 2). Next, the program has evaluated the function, using the initial values, and has printed out the current value. Notice how much larger it is than the final value (Table 6). The co-ordinates are the current values of some of the parameters - the mean lengths can be found in that table. The gradient table gives the partial derivatives with respect to these parameters. Because the mean lengths are not permitted to vary in phase 1, the gradients for them are all zero. At the end of the phase (or after IPRINT iterations, whichever is sooner) these statistics are printed again. In the example, phase 1 ended after 13 iterations, which required 20 function evalutations; the function having been reduced to 130.81. IEXIT shows how the run was terminated, and can be used in case of trouble.

The serious user should experiment from this point, using this example or a set of simulated data of his own; to study the effects of the input options on the program's ability to find reasonable solutions. It should be possible to improve the estimate of Z. The input options we suggest varying are the number of age classes, the number of the first fully recruited age class, the penalties, the lower bound on the Brody coefficient, and the bounds on the standard deviations. Our own experiences with such experimentation are partly described in Fournier & Breen (1983).

5. GUIDELINES FOR RUNNING THE PROGRAM ON REAL DATA

From quite a messy-looking set of simulated length frequencies (Figure 1), the program found a reasonable set of parameter values (Table 6). Analysis of real data is usually more difficult, because the user might know none of the 'true' parameter values and because real data never meet all the

model assumptions as nicely as simulated data. Where should the user go with this procedure after obtaining a solution?

There are no established rules. Schnute & Fournier (1980) describe in some detail the reasoning behind an analysis of two sets of real data with a program similar to this one. The user should read their conclusions as well as the guidelines below.

The first point is that the penalties and bounds, together with the various other input options, make it possible for a user to obtain almost any solution he wants, if he wants it badly enough and doesn't mind if the objective function is high. To obtain a meaningful answer, the user must develop for himself and then follow some procedure that leads him to a reasonable solution. It is for this reason that we suggest setting initially wide bounds on the mean lengths, population proportions and standard deviations; and setting a low bound on the Brody coefficient. We further suggest that the user then leave the bounds alone unless he has a specific reason to change one.

Second, there are many possible competing solutions. The user may want to look at the objective functions to compare them, but he should remember that his own goal is not to find a solution with a low objective function - it is instead to find good estimates for growth and mortality rates. Some solutions, even though they may have low objective functions, might contain estimates which are unsatisfactory - an estimate of the first mean length which is too low, for instance; or standard deviations which are too large; or a Brody coefficient which is too small. Our second suggestion is that the user should use his options to force biological realism where he has to. In particular, he can usually obtain some independent estimate of the Brody coefficient, and then set the lower bound accordingly.

Third, if competing solutions nevertheless give similar growth and mortality estimates, then those estimates are 'robust'. Fournier & Breen (1983) give an example where the number of age classes at one site (Hoskins Islet) was totally unknown. The program was run using a range of age classes, and the estimated total mortality rate varied only a small amount. The only way to discover that an estimate is robust is to obtain the same estimate over a range of program input conditions. We suggest the user do that in finding his final solution. If it turns out that the estimates are highly sensitive to changes in parameter values, that means the data don't contain much information about growth and mortality.

Fourth, in choosing an end solution, the user should be able to rely on his experiences with simulated data. So the user should spend some time 'playing' with the model, using a set of simulated data, to see how performance is varied with the different input options.

The user may wonder whether the solution he obtains is unique; in other words whether there may be a very different combination of parameters that fit the data equally well or better. Apart from running the program through the whole hyperspace of input parameter combinations, there isn't any way to be sure that multiple local solutions don't exist. If the user wants to experiment, he should use radically different initial values (the initial estimates of mean lengths, standard deviations, Brody coefficient, and

population proportions), to try to bump the model over the hill between alternative solutions. If you find a case of widely different solutions using the same input options but different initial values, please let us know.

6. BASIC TROUBLESHOOTING

A number of things can go wrong, and the authors probably haven't even seen them all. Below are some of the more common errors, which will be errors no matter what system the program is run on. The error messages given here are specific to the Vax at PBS, but the non-Vax user might get vaguely analogous messages.

1. Input data file not assigned correctly. There has to BE an input data file, and it has to be assigned to unit 105. If the program cannot find the file, the PBS operating system writes:

```
%FOR-F-FILNOTFOU, file not found
unit 105  file DISK$USER:[TROLL]FOR105.DAT;
user PC 000057DC
-RMS-E-FNF, file not found
```

2. Insufficient input values in the data file. If the program tries to read more input values than there are, the operating system writes:

```
%FOR-F-ENDDURREA, end-of-file during read
unit 105  file DISK$USER:[TROLL]INVB.DAT;145
user PC 000059D6
-RMS-E-EOF, end of file detected
%TRACE-F-TRACEBACK, symbolic stack dump follows
```

Usually, in this case, the number of age classes was greater than the number of input values for bounds - perhaps you decided to try 11 age classes instead of 9, and forgot to modify the input file.

3. Initial parameter estimates out of bounds. In the input file, you put bounds on the Brody coefficient, the first and last standard deviations of length at age, the mean lengths, and the population proportions. You cannot then give the program an initial parameter value which is outside those bounds. The program checks the initial values against the bounds, and prints error messages at the top of the main output file:

```
ERROR: INITIAL BRODY COEFFICIENT BENEATH BOUND
ERROR: FIRST OR LAST MEAN OUTSIDE THE BOUNDS
ERROR: POPULATION PROPORTION 6 OUT OF BOUNDS
ERROR: FIRST STANDARD DEVIATION OUTSIDE BOUNDS
```

Depending on the error, the program will then stop after printing out the initial values, or it will abort after printing out only some initial values. The error message will look like this:

```
%MTH-F-LOGZERNEG, logarithm of zero or negative value
  user PC 00005AC3
%TRACE-F-TRACEBACK, symbolic stack dump follows
module name     routine name                    line      rel PC     abs
                                                          00023900   00023900
                                                          00020274   00020274
GETINPUT        GETINPUT                         96       000003EF   00005AC3
TEST$MAIN       TEST$MAIN                        40       00000098   00003C98
```

Where the problem occurs in subroutines GETINPUT or POPEST, the problem lies somewhere in the input values, so check for error messages at the top of the output file.

PBS VAX USERS:

If the program is being run from batch (which is the more efficient way to run it), error messages will appear on the batch job log file - their subsequent handling will depend on the form of the batch SUBMIT command. If the job is run interactively, however, these messages will be ephemeral. This is another advantage to running the program from batch.

4. Population proportions sum to more than 1. An error message will appear at the head of the main output file:

```
ERROR: POPULATION PROPORTIONS SUM TO MORE THAN 1.0
```

5. Free-format syntax error. See your Fortran manual for proper formats.

```
%FOR-F-LISIO_SYN, list-directed I/O syntax error
unit 105  file DISK$USER:[BB26.V.TEST]INVB.DAT;151
```

6. Arithmetic fault, floating overflow. Fournier & Breen (1983) ran the program in many sets of 100 runs each, each run operating on simulated data generated with a new seed. At a normal frequency of about 1%, the run terminated with an error message of the form:

```
%SYSTEM-F-FLTOVF_F, Arithmetic fault, floating overflow at ....
```

These usually originated in UDFMIN, less commonly in CALCFG. These are a different class of 'error' from those above, because they occur only in some rare combinations of input. The problem is caused by weird conditions that lead the program to take a bad step in trying to minimize the function, and they may involve limits of precision. The frequency of the problem was highest when we set very tight bounds on the standard deviations of the lengths around their means. If the user encounters this problem, he should experiment with the input options.

7. SIMULATOR

The source code for a length frequency simulator is presented in Appendix 2. The simulator creates a set of length frequencies determined by a set of growth parameters and a total mortality rate specified by the user. Several sources of variation are introduced: first, the user specifies the standard deviations of the lengths at age around their means. Second, the user specifies the variance of the population proportions (how they vary from the ideal values determined by the total mortality rate). Third, the user specifies variation in the mean lengths (how they vary from the ideal values determined by the von Bertalanffy parameters). Finally, the observations are produced by simulated sampling, which introduces sampling error. The user can control all four sources of error to produce whatever kind of length frequencies he wants.

The simulator is useful for determining how well the program can recover the growth and mortality parameters, in other words how well the program works under different conditions.

The input file for the simulator looks like this:

```
SIMULATION FOR TROLL'S GUIDE
1500
8  1 49
140 0.25 0.0 5.0 0.2
0.25
2.0  11197
.3  .25
```

The first record is a title, which will appear on one of the output files. The second record is the value for NFISH, the number of sample observations to be simulated. There is no limit to this, except that very large values may cause overflows of the formats where the values are output. Sampling error decreases as sample size increases.

The third record contains NAGE, the number of age classes to be simulated (maximum 10); MINAGE, the age of the first age class: and NFREQ, the number of observed length intervals to be simulated (maximum 100).

The fourth record contains the three von Bertalanffy parameters L infinity, Brody coefficient and t zero, and the parameters a and b which determine the standard deviations of the lengths at age (equation 3).

The fifth record contains IARR(15), the permitted standard deviation of the means around the von Bertalanffy curve. Some users are confused by the existence of two kinds of standard deviation. In the previous record we specified the standard deviation of the lengths of fish of a given age around the mean length at age. Here we are specifying the standard deviation of the mean lengths around their values on a perfect von Bertalanffy curve. For a given set of the three von Bertalanffy parameters, the mean length for a given age is determined. The simulator randomly varies the mean lengths at age around these values, with standard deviation IARR(15):

$$\ell_i = L_\infty (1-e^{-K(i-t_0)})\ e^{\varepsilon i} \qquad (4)$$

where ℓ_i is mean length at age i and ε_i is a random normal deviate with mean zero and standard deviation of IARR(15).

The sixth record contains RANGE, which is the number of standard deviations below the first mean length and above the last mean length which will be simulated, and ISEEDX, which is a seed for the random number function. This latter value should be a large odd integer, and should be changed between simulations. If RANGE is small, the actual simulated sample will be somewhat smaller than that specified by NFISH.

The last record contains ZMORT, the total mortality rate, and RSIG, which determines the variation in the population proportions. RSIG is analagous with IARR(15), and is the standard deviation of T in:

$$P_i = P_1\ e^{\ T_i - Zi}$$

where P_i is the relative abundance of individuals at age i, P_1 is the abundance in age class 1, Z is total mortality rate, and T_i is a random normal deviate with mean zero and standard deviation RSIG.

In the example here, we simulate a population with 8 age classes, L infinity = 140, Brody coefficent = .25, t zero = 0, variance of the mean lengths 2, the a and b parameters 5.0 and 0.2. We will take 1500 simulated observations, distributed among 49 length intervals. The input file for this is above.

PBS VAX USERS: Use the following LINK command:
$LINK NASIM,SYS$SHARE:IMSLIBS/LIB.
Then assign your input file to unit 10 and if you wish
assign output file names to units 20 and 21.
$RUN NASIM.

OTHER USERS:
Unless you have the IMSL library, you will have to obtain code to generate random numbers with mean zero and specified standard deviations. This code should replace calls to the function GGUBFS in the simulator. Input must be on unit 10, and the program will produce output files on units 20 and 21.

ALL USERS:

The output from this example is shown in Tables 8 and 9. In the file on unit 20, the simulator first echoes back the input, so that it can be checked. It then shows the mean lengths, standard deviations, 'true' population percentages and the simulated population percentages (they differ because of sampling error), and the length frequencies.

Because of the error introduced by RSIG, no program could ever estimate perfectly the simulated mortality rate ZMORT. The best estimate one could obtain, if one had perfect ageing, is calculated by the simulator and printed out:

THE ESTIMATED TOTAL MORTALITY RATE WITH PERFECT AGEING IS 0.245

When these simulated data are analysed, the program is doing well if it recovers this estimate of mortality.

The output file on unit 21 (Table 8) contains the length frequencies and the information required to read them.

8. ACKNOWLEDGEMENTS

We thank Vivian Haist and an anonymous reviewer who provided helpful comments on the manuscript; and Wolfgang Carolsfeld, who drew the figures for us.

9. LITERATURE CITED

Fournier, D.A., & P.A. Breen. 1983. Estimation of abalone mortality rates with growth analysis. Trans. Am. Fish. Soc. 112:403-411.

Schnute, J., & D.A. Fournier. 1980. A new approach to length-frequency analysis: growth structure. J. Fish. Res. Bd. Can. 37:1337-1351.

Ricker, W.E. 1975. Computation and interpretation of biological statistics of fish populations. Bull. Fish. Res. Bd. Can. 191:382 pp.

Table 1. A file of input data for the program.

```
10 400  2
0.5  1.
30
          49              1   20.85287          2.269705
   18.0   38.0   65.0   76.0   80.0   91.0   50.0   48.0   29.0   16.0
   10.0   16.0   18.0   34.0   34.0   32.0   36.0   23.0   15.0   23.0
   22.0   34.0   23.0   30.0   33.0   25.0   31.0   19.0   30.0   28.0
   19.0   27.0   31.0   25.0   35.0   32.0   31.0   37.0   33.0   23.0
   33.0   22.0   36.0   22.0   21.0   11.0    8.0    7.0    3.0
.20  1  1.5  7.5   2.0  10.0
10*0.
10*1.
10*0.
10*200.
32  120  0.31  4.5  1.0
0.25 0.15  0.13  0.11  0.09  0.07  0.06  0.05  0.04 0.03
FIRST RUN FOR USERS' GUIDE USING SIMULATED DATA
```

Table 2. The first page of output from the program, using the example discussed in the text. This is printed at the beginning of the first phase.

```
22-AUG-83 10:29:57

  YOUNGEST FULLY RECRUITED AGE CLASS IS     2
INSTANTANEOUS NAT MORT IS .50 PENALTY IS 1.0

 NUMBER OF AGE CLASSES  10   NUMBER OF VARIABLES  26
  MAXIMUM NUMBER OF FUNCTION EVALUATIONS  400
    THE NUMBER OF ITERATIONS PER PRINTOUT IS  30
   THE NUMBER OF LENGTH FREQUENCIES IS  49   YOUNGEST AGE CLASS   1
  SHORTEST LENGTH  20.85   INTERVAL LENGTH  2.27

-   THE LENGTH FREQUENCIES ARE

 18.0  38.0  65.0  76.0  80.0  91.0  50.0  48.0  29.0  16.0
 10.0  16.0  18.0  34.0  34.0  32.0  36.0  23.0  15.0  23.0
 22.0  34.0  23.0  30.0  33.0  25.0  31.0  19.0  30.0  28.0
 19.0  27.0  31.0  25.0  35.0  32.0  31.0  37.0  33.0  23.0
 33.0  22.0  36.0  22.0  21.0  11.0   8.0   7.0   3.0
LOWER BOUND ON BRODY COEFF.:  0.200
VARIANCE OF MEANS AROUND VON BERTALANFFY:  1.00
BOUNDS ON FIRST AND LAST STANDARD DEVIATION:     1.50   7.50   2.00  10.00

LOWER AND UPPER BOUNDS ON THE PROPORTIONS ARE:

  0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
  1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00

LOWER AND UPPER BOUNDS ON THE MEAN LENGTHS ARE:

  0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
200.00 200.00 200.00 200.00 200.00 200.00 200.00 200.00 200.00 200.00

INITIAL X VALUES ARE:

 32.00 120.00   0.31   4.50   1.00   0.00

THE INITIAL LENGTHS ARE:

 32.00  56.99  75.32  88.77  98.63 105.86 111.16 115.05 117.91 120.00

THE INITIAL ESTIMATES FOR THE POPULATION PROPORTIONS ARE:

  0.25   0.15   0.13   0.11   0.09   0.07   0.06   0.05   0.04   0.05
THE TOTAL NUMBER OF FISH IS 1483.
```

Table 3. Second page of output from the program, using the example discussed in the text. This is printed at the end of the first phase.

```
FIRST RUN FOR USERS' GUIDE USING SIMULATED DATA

THE VON BERTALANFFY PARAMETERS:
LINFINITY= 125.76     K=  0.310     TZERO=  0.053

THE ESTIMATES FOR THE MEAN LENGTHS ARE:
 32.00  56.99  75.32  88.77  98.63 105.86 111.16 115.05 117.91 120.00

ESTIMATES FOR THE POPULATION PROPORTIONS ARE:
 0.348  0.159  0.135  0.083  0.078  0.066  0.031  0.028  0.031  0.040

THE ESTIMATED TOTAL MORTALITY RATE IS   0.219

THE ESTIMATED STANDARD DEVIATIONS OF THE LENGTHS AROUND THE MEAN LENGTHS ARE:
  4.50   4.83   4.97   5.08   5.17   5.25   5.32   5.38   5.44   5.50

ESTIMATED VARIANCE OF MEAN LENGTHS AROUND
THE VON BERTALANFFY CURVE IS  0.00000

THE OBJECTIVE FUNCTION IS    130.81
```

Table 4. Third page of output from the program, using the example discussed in the text. This is printed at the end of the second phase.

```
FIRST RUN FOR USERS' GUIDE USING SIMULATED DATA

THE VON BERTALANFFY PARAMETERS:
LINFINITY= 132.37     K=  0.243     TZERO= -0.133

THE ESTIMATES FOR THE MEAN LENGTHS ARE:
 31.64  53.93  70.66  83.40  94.41 102.52 108.93 114.08 118.02 121.01

ESTIMATES FOR THE POPULATION PROPORTIONS ARE:
 0.342  0.142  0.114  0.095  0.069  0.085  0.043  0.037  0.040  0.032

THE ESTIMATED TOTAL MORTALITY RATE IS   0.190

THE ESTIMATED STANDARD DEVIATIONS OF THE LENGTHS AROUND THE MEAN LENGTHS ARE:

  4.50   4.83   4.97   5.08   5.17   5.25   5.32   5.38   5.44   5.50

ESTIMATED VARIANCE OF MEAN LENGTHS AROUND
THE VON BERTALANFFY CURVE IS  0.04566

THE OBJECTIVE FUNCTION IS    79.078
```

Table 5. Fourth page of output from the program, using the example discussed in the text. This was printed at the end of the third phase.

```
FIRST RUN FOR USERS' GUIDE USING SIMULATED DATA

 THE VON BERTALANFFY PARAMETERS:
 LINFINITY= 129.06     K=  0.265     TZERO= -0.068

 THE ESTIMATES FOR THE MEAN LENGTHS ARE:
  31.72  54.53  71.90  84.87  95.46 103.07 109.15 113.91 117.45 120.03

 ESTIMATES FOR THE POPULATION PROPORTIONS ARE:
  0.344  0.145  0.123  0.090  0.067  0.082  0.037  0.036  0.042  0.034

 THE ESTIMATED TOTAL MORTALITY RATE IS   0.191

 THE ESTIMATED STANDARD DEVIATIONS OF THE LENGTHS AROUND THE MEAN LENGTHS ARE:
   5.11   5.45   5.59   5.69   5.78   5.86   5.93   6.00   6.06   6.11

 ESTIMATED VARIANCE OF MEAN LENGTHS AROUND
 THE VON BERTALANFFY CURVE IS  0.01364

 THE OBJECTIVE FUNCTION IS    59.172
```

Table 6. Final page of output from the program, using the example discussed in the text. This is printed at the end of the fourth and final phase.

```
FIRST RUN FOR USERS' GUIDE USING SIMULATED DATA

THE VON BERTALANFFY PARAMETERS:
LINFINITY= 130.13     K=  0.260     TZERO= -0.079

THE ESTIMATES FOR THE MEAN LENGTHS ARE:
 31.75  54.56  71.84  84.83  95.55 103.30 109.46 114.30 117.94 120.65

ESTIMATES FOR THE POPULATION PROPORTIONS ARE:
 0.345  0.145  0.122  0.091  0.069  0.083  0.039  0.036  0.039  0.033

THE ESTIMATED TOTAL MORTALITY RATE IS   0.197

THE ESTIMATED STANDARD DEVIATIONS OF THE LENGTHS AROUND THE MEAN LENGTHS ARE:
  5.17   5.33   5.40   5.45   5.50   5.54   5.57   5.60   5.63   5.66

ESTIMATED VARIANCE OF MEAN LENGTHS AROUND
THE VON BERTALANFFY CURVE IS  0.01735

THE OBJECTIVE FUNCTION IS    58.808

    ALL PHASES COMPLETED
```

Table 7. The first page of the statistical report.

```
 22-AUG-83 10:29:57
1INITIAL STATISTICS :
+_____________

 ITER. NO.      0   FCN EVALUATIONS      1  FCN VALUE:          229.3227081
 CO-ORDINATES
      32.0000000       88.0000000        0.3768586        1.7320508        1.0000000
       0.0000000       32.0000000       56.9916916       75.3217697       88.7659073
      98.6264725      105.8586731      111.1631088      115.0536270      117.9071198
     120.0000000       -1.3862944       -1.8971200       -2.0402210       -2.2072749
      -2.4079456       -2.6592600       -2.8134108       -2.9957323       -3.2188759
      -2.9957314
 GRADIENT
       0.0000000        0.0000000        0.0000000        0.0000000        0.0000000
       0.0000000        0.0000000        0.0000000        0.0000000        0.0000000
       0.0000000        0.0000000        0.0000000        0.0000000        0.0000000
       0.0000000     -288.8199158      -30.5917377       -3.4575949       59.1710052
      45.4167137       44.2882347       51.4754257       45.5565338       34.6752701
      42.2860680
-FINAL STATISTICS :
+_____________

   ITERATION NO.      13     FUNCTION EVALUATIONS      20     EXIT CODE   1
 FUNCTION VALUE=  0.130808715820D+03
 CO-ORDINATES
  0.320000000000D+02  0.880000000000D+02  0.376858562231D+00  0.173205077648D+01  0.100000000000D+01  0.000000000000D+00
  0.320000000000D+02  0.569916915894D+02  0.753217697144D+02  0.887659072876D+02  0.986264724731D+02  0.105858673096D+03
  0.111163108826D+03  0.115053627014D+03  0.117907119751D+03  0.120000000000D+03 -0.105400598049D+01 -0.183510077000D+01
 -0.199934208393D+01 -0.248821663857D+01 -0.254716440016D+01 -0.271417903900D+01 -0.347262763977D+01 -0.356978869438D+01
 -0.348227667809D+01 -0.323000502586D+01
 GRADIENT
  0.000000000000D+00  0.000000000000D+00  0.000000000000D+00  0.000000000000D+00  0.000000000000D+00  0.000000000000D+00
  0.000000000000D+00  0.000000000000D+00  0.000000000000D+00  0.000000000000D+00  0.000000000000D+00  0.000000000000D+00
  0.000000000000D+00  0.000000000000D+00  0.000000000000D+00  0.000000000000D+00  0.239330142736D+00 -0.350428223610D-01
 -0.614910386503D-01 -0.287566520274D-01  0.860632599820-01  0.820476561785D-03  0.168057113886D+00  0.767602026463D-01
 -0.956872776151D-01 -0.811216458678D-01
 RETURNED FROM DFMIN  -  IEXIT =    1
```

Table 8. Output from the simulator.

SIMULATION FOR TROLL'S GUIDE

THE SEED FOR RANDU IS 11197.

TOTAL MORTALITY RATE IS 0.30
STANDARD DEVIATION FOR RECRUITMENT IS 0.25

THE ESTIMATED TOTAL MORTALITY RATE WITH PERFECT AGEING IS 0.245

L INFINITY	140.000
T ZERO	0.000
VB K (Brody coefficient)	0.250
A PARAMETER	5.000
B PARAMETER	0.200

AGE	MEAN	STD DEV	TRUE %	SIMULATED %
1	31.25	5.20	34.31	38.57
2	54.87	5.28	14.66	16.43
3	73.89	5.35	12.69	14.08
4	88.07	5.40	10.76	11.66
5	99.82	5.45	8.14	8.87
6	108.83	5.49	9.46	10.49
7	115.89	5.53	5.89	5.87
8	120.94	5.57	4.09	4.03

LENGTH	FREQUENCY
21.99	18.0
24.26	38.0
26.53	65.0
28.80	76.0
31.07	80.0
33.34	91.0
35.61	50.0
37.88	48.0
40.15	29.0
42.42	16.0
44.68	10.0
46.95	16.0
49.22	18.0
51.49	34.0
53.76	34.0
56.03	32.0
58.30	36.0
60.57	23.0
62.84	15.0
65.11	23.0
67.38	22.0
69.65	34.0

Table 8 (cont'd)

LENGTH	FREQUENCY
71.92	23.0
74.19	30.0
76.46	33.0
78.73	25.0
81.00	31.0
83.27	19.0
85.54	30.0
87.81	28.0
90.08	19.0
92.35	27.0
94.62	31.0
96.89	25.0
99.16	35.0
101.43	32.0
103.70	31.0
105.97	37.0
108.24	33.0
110.51	23.0
112.78	33.0
115.05	22.0
117.32	36.0
119.59	22.0
121.85	21.0
124.12	11.0
126.39	8.0
128.66	7.0
130.93	3.0

Table 9. Output from the simulator, designed to be used as input to the program.

	49		1	20.85287		2.269705			
18.0	38.0	65.0	76.0	80.0	91.0	50.0	48.0	29.0	16.0
10.0	16.0	18.0	34.0	34.0	32.0	36.0	23.0	15.0	23.0
22.0	34.0	23.0	30.0	33.0	25.0	31.0	19.0	30.0	28.0
19.0	27.0	31.0	25.0	35.0	32.0	31.0	37.0	33.0	23.0
33.0	22.0	36.0	22.0	21.0	11.0	8.0	7.0	3.0	

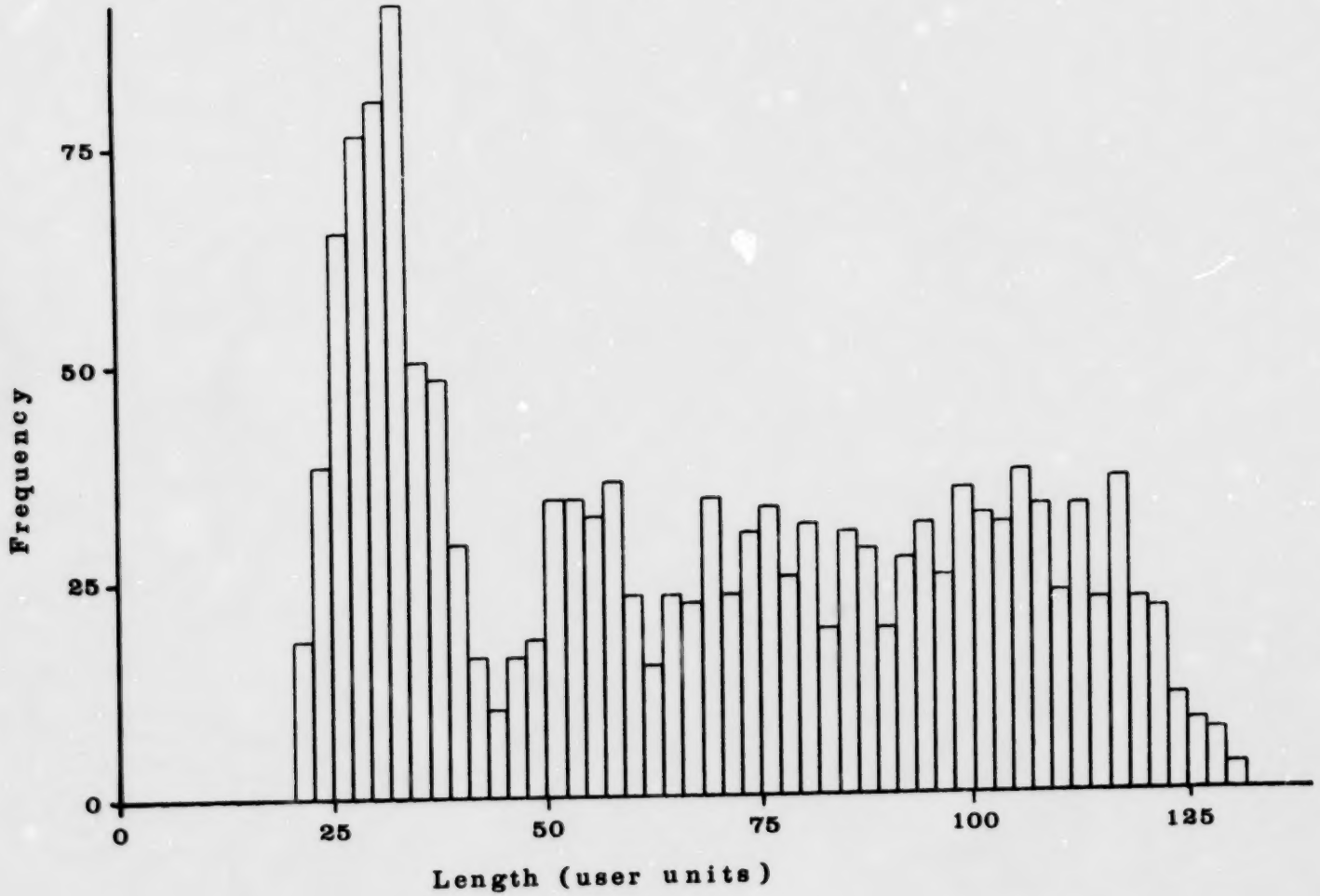

Fig. 1. Size frequency diagram, drawn from simulator results used in the text examples.

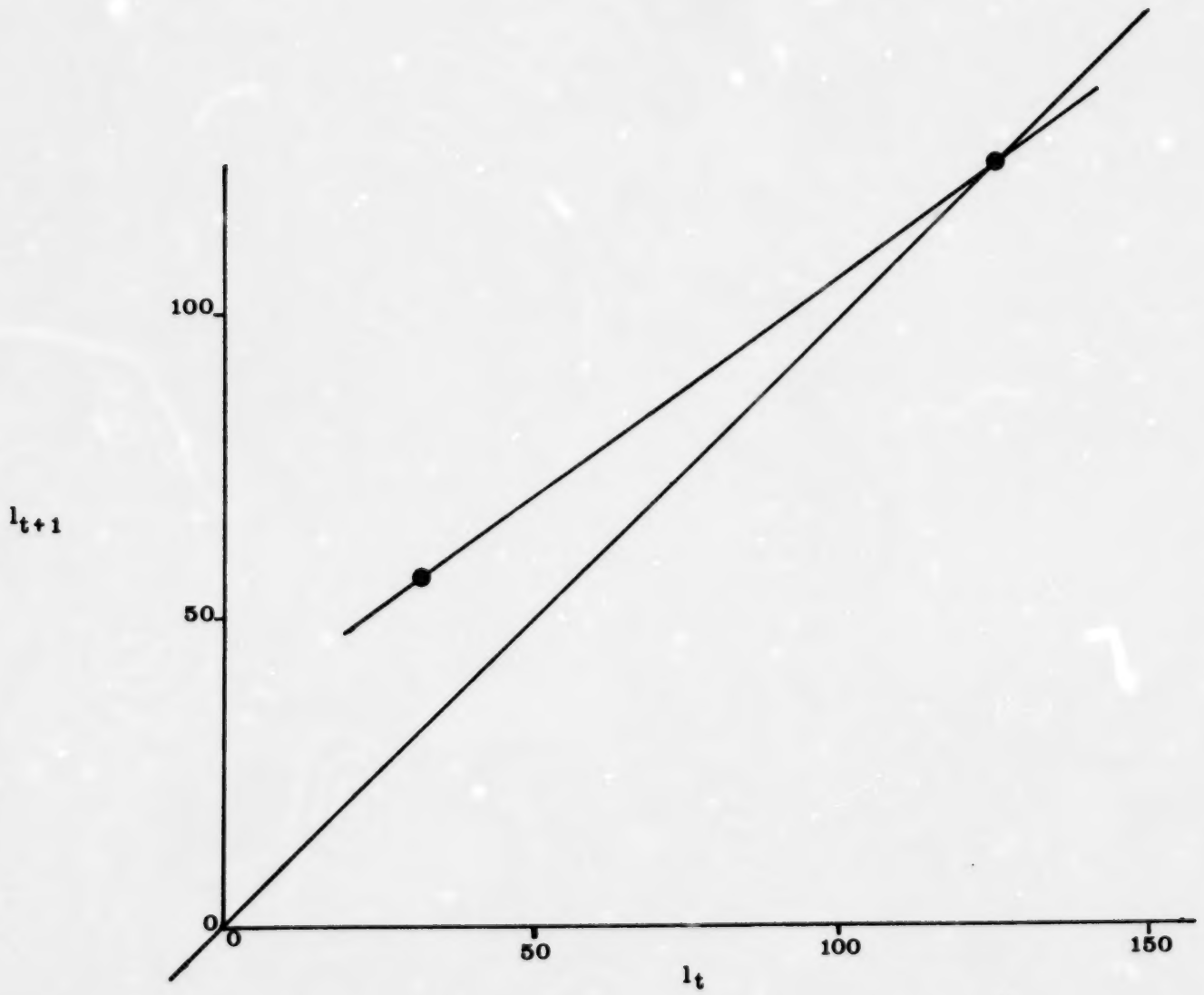

Fig. 2. See text for details.

APPENDIX 1. Fortran source code for the program described in this report.

```
C*****************************************
C
C PROGRAM FOR THE MODEL DESCRIBED BY BREEN & FOURNIER (1983)
C  SEE TRANS. AM. FISH. SOC. 112:403-411.
C
C TABLE OF VARIABLES TO BE INPUT ON UNIT 105:
C
C NAME      TYPE          MEANING
C-----------------------------------------------------------------
C FIRST RECORD:
C NK        INTEGER     NUMBER OF AGE CLASSES
C MAXFN     INTEGER     MAXIMUM NUMBER OF FUNCTION EVALUATIONS
C NFULL     INTEGER     THE NUMBER OF THE FIRST FULLY-RECRUITED
C                       AGE CLASS
C
C SECOND RECORD:
C FMORT     REAL        DETERMINES WHETHER TOTAL MORTALITY
C                            IS CALCULATED BY THE PROGRAM
C PENMOR    REAL        THE 'PENALTY' ASSOCIATED WITH POPULATION
C                      PROPORTIONS THAT DO NOT DECREASE EXPONENTIALLY
C
C THIRD RECORD:
C IPRINT    INTEGER     MAXIMUM NUMBER OF ITERATIONS PER PRINTOUT
C
C FOURTH RECORD:
C N5        INTEGER     NUMBER OF LENGTH FREQUENCY OBSERVATIONS
C W5.       INTEGER     THE AGE OF THE FIRST AGE CLASS
C S5        REAL        THE LOWER EDGE OF THE LENGTH INTERVAL OF THE
C                            FIRST FREQUENCY OBSERVATION
C S6        REAL        WIDTH OF EACH LENGTH INTERVAL
C
C FIFTH RECORD AND SUBSEQUENT FEW:
C FREQ      REAL        LENGTH FREQUENCY OBSERVATIONS.  EACH NUMBER IS
C                            THE NUMBER OF INDIVIDUALS OBSERVED TO FALL
C                            WITHIN A GIVEN LENGTH INTERVAL.
C NEXT RECORD:
C FKMIN     REAL        LOWER BOUND ON bRODY COEFFICIENT
C PP        REAL        'PENALTY' ASSOCIATED WITH VARIANCE
C                      OF MEAN LENGTHS AROUND THE VON BERTALANFFY CURVE
C SDMIN     REAL        LOWER BOUND ON S.D. OF FIRST AGE CLASS
C V1MAX     REAL        UPPER BOUND ON S.D. OF FIRST AGE CLASS
C VKMIN     REAL        LOWER BOUND ON S.D. OF LAST AGE CLASS
C VKMAX     REAL        UPPER BOUND ON S.D. OF LAST AGE CLASS
C
C NEXT RECORD:
C PMIN(I),I=1,K REAL  LOWER BOUNDS ON PROPORTION OF EACH AGE CLASS
C
C NEXT RECORD:
C PMAX(I),I=1,K REAL  UPPER BOUNDS ON PROPORTION OF EACH AGE CLASS
C
C NEXT RECORD:
```

```
C
C NEXT RECORD:
C FMMIN(I),I=1,K REAL LOWER BOUNDS ON THE MEAN LENGTH OF EACH AGE CLASS
C
C NEXT RECORD:
C FMMAX(I),I=1,K REAL UPPER BOUNDS ON THE MEAN LENGTH OF EACH AGE CLASS
C
C NEXT RECORD:
C X(1)    REAL        MEAN LENGTH ESTIMATE FOR FIRST AGE CLASS
C X(2)    REAL        MEAN LENGTH ESTIMATE FOR THE LAST AGE CLASS
C X(3)    REAL        ESTIMATE FOR THE BRODY COEFFICENT
C X(4)    REAL        ESTIMATED S.D. FOR THE FIRST AGE CLASS
C X(5)    REAL        DIFFERENCE BETWEEN THE SD OF FIRST AND LAST AGE CLASSES
C
C NEXT RECORD:
C X(J)    REAL        INITIAL VALUES FOR POPULATION PROPORTIONS
C
C NEXT RECORD:
C TITLE(I) CHARACTER  TITLE CARD
C-------------------------------------------------------------------------
C  ****************************************
      CHARACTER T*8,D*9
      DIMENSION X(40), G(40), EPS(40), H(820)
      REAL*4 M(20)
      REAL*4 F,FEST
      INTEGER W5
      LOGICAL LLOG
      INTEGER TITLE(40)
      DIMENSION FREQ(85)
      DIMENSION FMMIN(20), FMMAX(20)
      DIMENSION GEST(10)
      DIMENSION W(160)
      DIMENSION BBETA1(20)
      DIMENSION PMIN(20) ,PMAX(20)
      DIMENSION CC(20), OMEGA(20)
      DIMENSION PR(1,1)
      DIMENSION JMIN(1), JMAX(1)
      DATA DFN/1.D3/, MAXFN/250/,IPRINT/1/,MODE/1/
C
C   THIS STAMPS THE TIME AND DATE ON BOTH OUTPUT FILES
      CALL DATE (D)
      CALL TIME (T)
      WRITE (108,73) D,T
      WRITE (109,73) D,T
73    FORMAT(1X,A9,1X,A8)
C
      NPAR=6
      IFN=0
      IPFLAG=0
      IPHASE=0
      IRETURN=0
      ITRY=1
      II13=2
C
```

```
      CALL GETINPUT( N,X,NPAR,FREQ,PMIN,PMAX,IMODE,MAXFN,IPRINT,
     *  IMAX,N5,W5,S5,S6, FLMIN, FLMAX, FHMIN, FHMAX, FKMIN,
     *  PP,SDMIN, SDMIN1, FTOT, TITLE  ,V1MAX, VKMAX  ,VKMIN, FMMIN,
     *   FMMAX ,OMEGA ,CC,BBETA,BBETA1, VFLAG,PR12M, PRK2M,
     *   PRKLM, PRKUM ,PR,JMIN,JMAX,FREQTO,NTRY,II17,
     *   FMORT,NFULL,PENMOR )
C
      K=(N-NPAR)/2
      UMIN=EXP(FKMIN)
   25 CONTINUE
      CALL EPSSET(N,EPS,IPHASE,K)
      CALL SSET(IPHASE,II0,II2,II5,II8,II11)
    8 CONTINUE
   10 CONTINUE
      CALLODFMIN(F,X,G,N,W,DFN,MAXFN,IPRINT,EPS,IEXIT,
     *   IMAX,IMODE,H,MODE ,N1,IJ,JK,DF,ICON,NN,I,IK, ICOUNT,FY,IS,
     *   J,II,GS,GYS, LLOG,IU,Z,K,GSO, DGS,IFN,IV,I1,DMIN,
     *   ALPHA,LINK,NP,IB,ZZ,ITN,TOT,SIG,IRETURN,IHANG)
      IF ( IRETURN .EQ. 1 ) GO TO 70
      IF ( IRETURN .EQ. 2 ) GO TO 80
      IF ( IRETURN .EQ. 3) GO TO 90
      IF (IRETURN .EQ. 4) GO TO 90
      IF (IRETURN .EQ. 5) GO TO 100
   70 CONTINUE
      CALL CALCFG(N, ,F,G,NPAR,FREQ,PMIN,PMAX,
     *  IMAX,N5,W5,S5,S6, FLMIN, FLMAX, FHMIN, FHMAX, UMIN,
     *  PP, SDMIN, SDMIN1,FTOT ,V1MAX , VKMAX  , VKMIN ,FMMIN, FMMAX ,
     *   OMEGA, CC, BBETA , BBETA1 , VFLAG , PR,JMIN, JMAX ,
     *   FREQTO,PR12M,PRK2M,PRKLM,PRKUM,II0,II1,II2,II3,II4,II5,II6,II7,
     *   II8,II9,II10,II11,II12,II13,II14,II15,IPHASE,
     *   II17,FMORT,NFULL,PENMOR )
      GO TO 10
   80 CONTINUE
      CALL CALCFG(N,X,FY,W, NPAR,FREQ,PMIN,PMAX,
     *  IMAX,N5,W5,S5,S6, FLMIN, FLMAX, FHMIN, FHMAX, UMIN,
     *  PP, SDMIN, SDMIN1,FTOT ,V1MAX , VKMAX  , VKMIN ,FMMIN, FMMAX ,
     *   OMEGA, CC, BBETA , BBETA1 , VFLAG , PR,JMIN, JMAX ,
     *   FREQTO,PR12M,PRK2M,PRKLM,PRKUM,II0,II1,II2,II3,II4,II5,II6,II7,
     *   II8,II9,II10,II11,II12,II13,II14,II15,IPHASE,
     *   II17,FMORT,NFULL,PENMOR )
      GO TO 10
   90 WRITE(109,95) IEXIT
   95 FORMAT( ' RETURNED FROM DFMIN  -  IEXIT =', I5 )
      CALL FINPRT(K,X,F,G,NPAR,TITLE, SDMIN, SDMIN1,FKMIN,NFULL,PENMOR,
     *  UMIN,W5,FLMIN,FLMAX,FHMIN,FHMAX,PP)
      IF (IRETURN.EQ.3) CALL EXIT
      IPHASE=IPHASE+1
      IF (IPFLAG .EQ. 1) IPHASE=IPHOLD
      IPFLAG=0
      CALL SSET(IPHASE,II0,II2,II5,II8,II11)
      CALL EPSSET(N,EPS,IPHASE,K)
      IRETURN=0
      IF (1 .EQ. 1) GO TO 8
      CALL EXIT
```

```
  100 IPHOLD=IPHASE
      IPHASE=0
      IPFLAG=1
      IRETURN=0
      WRITE(108,110)
  110 FORMAT(/,5X,'RESTART ATTEMPT')
      GO TO 25
      END
C ******************
C  SUBROUTINE CALCFG
C ******************
      SUBROUTINE CALCFG(N,X,F,G,NPAR,FREQ,PMIN,PMAX,
     *  IMAX,N5,W5,S5,S6, FLMIN, FMIN,FMAX,FLMAX,UMIN,
     *  PP, SDMIN, SDMIN1,FTOT ,V1MAX , VKMAX  , VKMIN ,FMMIN, FMMAX ,
     *   OMEGA, CC, BBETA , BBETA1 , VFLAG , PR,JMIN, JMAX ,
     *   FREQTO,PR12M,PRK2M,PRKLM,PRKUM,II0,II1,II2,II3,II4,II5,II6,II7,
     *   II8,II9,II10,II11,II12,II13,II14,II15,IPHASE,
     *   II17,FMORT,NFULL,PENMOR )
      DIMENSION TU(20), TL(20)   , SE4(20) , FMMIN(1), FMMAX(1)
      DIMENSION E4(20),E4L(20),E4U(20),Q1L(20),Q1U(20),USUM(20),
     1  DUSUM(20)
      INTEGER A
      INTEGER  W5
      REAL P9
      REAL*4 L,F
      REAL*4 MM(20)
      DIMENSION T1(20),T2(20)
      REAL*4 T5(20), T6(20), M(20),
     * T(20), P(20), TEMP, L1(20), L2(20), L3(20),
     * Q1(20), S5,S6, Q,C8
      REAL*4 X(N),G(N)
      DIMENSION PMIN(20) ,PMAX(20)
      DIMENSION FPOW1(20)
      DIMENSION FREQ(1)
      DIMENSION T7(20)
      DIMENSION CC(1), OMEGA(1)
      DIMENSION BBETA1(1)
      DIMENSION PR(1,1)
      DIMENSION QA(10)
      DIMENSION TEMP4(20), DQ1S(20), DQ1M(20), JMIN(1),JMAX(1)
      DATA C7 /1./
C
      IF (IPHASE .GT. 1) GO TO 2010
      P9=1000.
      PE=10.
      Q9=100.
      GO TO 2025
C
 2010 IF (IPHASE .GT. 2) GO TO 2015
      P9=1000000.
      PE=100.
      Q9=100.
      GO TO 2025
C
```

```
 2015 P9=1000000.
      PE=1000.
      Q9=1000.
C
C  ***************************************************************************
C  CALCULATE VALUES IN TERMS OF SUBROUTINE VARIABLES
C  ***************************************************************************
C
 2025 K=(N-NPAR)/2
      DO 2 J=1,K
      FPOW1(J) = 2./(1. + BBETA1(J) )
    2 CONTINUE
      FM = FLMIN
      V=X(1)
      C=X(2)
      U=UMIN+X(3)**2
      UINV=1.0/U
      U5= SDMIN + X(4)**2
      IF ( VFLAG .LT. 0.D0 ) U6 = - X(5)**2
      IF ( VFLAG .EQ. 0.D0) U6 = X(5)
      IF ( VFLAG .GT. 0.D0 ) U6 = X(5)**2
 3005 CONTINUE
      FMU = FLMAX
      FMD = FMU - FM
      V1=U5
      VK=U5+U6
      ITEMP = NPAR + K
C
        DO 10 J=1,K
        P(J)=EXP(X(ITEMP+J))
   10   CONTINUE
C
      PSUM=0.0
        DO 14 J=1,K
        PSUM=PSUM+P(J)
   14   CONTINUE
C
       DO 16 J=1,K
        P(J)=P(J)/PSUM
   16   CONTINUE
C
      DO 20  J=1,K
         M(J)= X(NPAR+J)
         TEMP = FLOAT(J-1)/FLOAT(K-1)
         TEMP=SQRT(TEMP)
         T(J) = U5 + U6*TEMP
   20 CONTINUE
 3111 CONTINUE
C
      USUM(1)=0.0
      DO 4140 J=1,K-1
         USUM(J+1)=USUM(J)+UINV**(J-1)
 4140 CONTINUE
C
```

```
      DUSUM(1)=0.0
        DO 4150 J=1,K-1
        DUSUM(J+1)=DUSUM(J)+(J-1)*UINV**(J-2)
 4150   CONTINUE
C
        DO 4160 J=1,K
        MM(J)=V+C*USUM(J)/USUM(K)
 4160   CONTINUE
C
C  CALCULATE DERIVATIVES OF THE VARIANCES FOR THE CHAIN RULE.
C
        DO 4170 J=1,K
        TEMP=FLOAT(J-1)/FLOAT(K-1)
        TEMP=SQRT(TEMP)
        SIGN=1.0
        IF (T(J) .LT. 0.0) SIGN=-1.0
        T(J)=ABS(T(J))
        T5(J)=SIGN
        T6(J)=TEMP*SIGN
        T7(J)=TEMP*(TEMP-1.0)*SIGN
 4170   CONTINUE
C
C  ***************************************************************************
C  CALCULATE THE LOG LIKLIHOOD AND ITS DERIVATIVES WRT THE MEANS
C   AND VARIANCES  USING A MODIFIED HASSELBLAD METHOD.
C  ***************************************************************************
      L = 0.D0
      DO 90 J=1,K
         L1(J) = 0.D0
         L2(J)= 0.D0
         L3(J)=0.D0
   90 CONTINUE
      X5 = S5 - .5* S6
      INDEX = 0
      DO 100 I=1,N5
      Q = 0.D0
      X5 = X5 + S6
      IF ( (FREQ(I).EQ.0.) .AND. (II13.EQ.2) .AND. (II17.NE.1))
     *   GO TO 100
      DO 95 J=1,K
      FPOW= FPOW1(J)
      CC1 = CC(J)
      OMEGA1 = OMEGA(J)
      SE4(J)= (X5- M(J))/ T(J)
      Q1(J)= OMEGA1* EXP(-CC1*(ABS(SE4(J))) ** FPOW) * S6 / T(J)
      Q= Q + P(J)* Q1(J)
   95 CONTINUE
      Q2=Q
C
C  CALCULATE LIKLIHOOD CONTRIBUTION FOR LENGTH FREQUENCIES
C
      IF ( FREQ(I) .EQ. 0.D0 ) GO TO 991
      Q = Q* FTOT/ FREQ(I)
      IF ( Q .LE. 0.)    Q=1.E-20
```

```
      L = L + FREQ(I) * ALOG(Q)
  991 CONTINUE
      TEMP = FTOT/Q
      DO 998 J=1,K
      FPOW= FPOW1(J)
      CC1 = CC(J)
      SIGN = 1.
      IF ( SE4(J) .LT. 0.) SIGN = -1.
      AA = ABS(SE4(J))
      TEMP1= TEMP * P(J)
      DQ1M(J) = Q1(J) * CC1 * SIGN * FPOW * AA ** ( FPOW - 1. ) / T(J)
      DQ1S(J) = DQ1M(J) * SIGN * AA - Q1(J) / T(J)
  998 CONTINUE
C
      IF ( FREQ(I) .EQ. 0 ) GO TO 98
C
      DO 98 J=1,K
         TEMP1 = TEMP * P(J)
         L1(J) = L1(J) + TEMP1 * DQ1M(J)
         L2(J) = L2(J) + TEMP1 * DQ1S(J)
         L3(J)=L3(J)+TEMP*Q1(J)
   98 CONTINUE
C
  100   CONTINUE
C
C  INITIALIZE THE GRADIENT
C
      DO 50 J=1,N
      G(J) = 0.D0
   50 CONTINUE
C
C   SUGGESTED RATE OF DECLINE IN POPULATION
C
      IF ( FMORT .EQ. 0.) GO TO 8000
      FMORT=0.
      TEMP1=0.
      RBAR=FLOAT(K-NFULL)/2.
      PBAR=0.
      DO 504 J=NFULL,K
         PBAR=PBAR+ALOG(P(J))
  504 CONTINUE
      PBAR=PBAR/FLOAT(K-NFULL+1)
      DO 1007 J=NFULL,K
         FMORT=FMORT+(ALOG(P(J))-PBAR)*(J-NFULL-RBAR)
         TEMP1=TEMP1+(J-NFULL-RBAR)**2
 1007 CONTINUE
      FMORT=FMORT/TEMP1
      DO 1002 J=NFULL,K
         TEMP=ALOG(P(J))-PBAR-(J-NFULL-RBAR)*FMORT
         L=L-PENMOR*TEMP**2
         L3(J)=L3(J)-2.*PENMOR*TEMP /P(J)
         DO 202 JJ=NFULL,K
     L3(JJ)=L3(JJ)+2.*TEMP*(J-NFULL-RBAR)*(JJ-NFULL-RBAR)* PENMOR /
     *       ( P(JJ)* TEMP1 )
         L3(JJ)=L3(JJ)+2.*PENMOR*TEMP/FLOAT(K-NFULL+1)/P(JJ)
```

```
  202    CONTINUE
 1002 CONTINUE
 8000 CONTINUE
C
C  LIKLIHOOD CONTRIBUTION FOR  VON BERTALANFFY
C
      DO 105 J=1,K
       L=L-PP*(M(J)-MM(J))**2
  105 CONTINUE
C
C  ***************************************************************************
C  THIS SECTION COMPUTES THE PENALTY FUNCTION CONTRIBUTIONS TO THE
C  LIKLIHOOD FUNCTION FOR VIOLATING THE VARIOUS CONSTRAINTS.
C  ***************************************************************************
C
C  PENALTY FUNCTION CONTRIBUTIONS FOR VIOLATING CONSTRAINTS ON
C  THE POPULATION PERCENTAGES
C
      L=L-100.*ALOG(PSUM)**2
C
        DO 110 J=1,K
      IF(P(J) .LT. PMIN(J)) L=L+P9*(-100.*(P(J)-PMIN(J))**4)
      IF(P(J) .GT. PMAX(J)) L=L+P9*(-100.*(P(J)-PMAX(J))**4)
      IF(P(J) .LT. PMIN(J))L3(J)=L3(J)-P9*400.*(P(J)-PMIN(J))**3
      IF(P(J) .GT. PMAX(J))L3(J)=L3(J)-P9*400.*(P(J)-PMAX(J))**3
  110 CONTINUE
C
C  PENALTY FUNCTION CONTRIBUTION FOR VIOLATING ORDER OF MEAN LENGTHS.
C
      DO 140 I=1,K-1
      IF( M(I) .GT. M(I+1) ) L1(I+1) = L1(I+1) +P9*4.*
     *     ( M(I) - M(I+1)   ) **3
      IF ( M(I) .GT. M(I+1)  ) L1(I) =L1(I)-P9*4.*(M(I)-M(I+1))**3
      IF ( M(I) .GT. M(I+1)  )   L=L- P9   *(M(I) - M(I+1) )**4
  140 CONTINUE
C
C  PENALTY FUNCTION CONTRIBUTIONS FOR VIOLATING CONSTRAINTS
C  ON THE FIRST AND LAST VARIANCE.
C
      IF ( V1 .GT. V1MAX ) L=L- PE* ( V1 -V1MAX)**4
      IF ( VK .GT. VKMAX ) L=L-PE* ( VK - VKMAX )**4
      IF ( VK .LT. VKMIN) L=L - PE*( VK - VKMIN )**4
      IF ( V1 .GT. V1MAX ) G(4)= G(4) -4.*PE*( V1 - V1MAX )**3
      IF ( VK .LT. VKMIN)  G(4) = G(4) - 4.* PE* ( VK - VKMIN) **3
      IF ( VK .GT. VKMAX ) G(4)= G(4) - 4.*PE*( VK - VKMAX ) **3
      IF(( II2 .EQ.1 ).OR.(II5 .EQ.1)) GO TO 7144
      IF ( VK .LT. VKMIN) G(5) = G(5) -4.*PE*( VK - VKMIN )**3
      IF ( VK .GT. VKMAX ) G(5)= G(5) - 4. * PE * ( VK - VKMAX )**3
 7144 CONTINUE
      IF ( II4 .EQ. 1 ) GO TO 146
C
C  PENALTY FUNCTION CONTRIBUTION FOR VIOLATING CONSTRAINTS ON THE
C  UPPER AND LOWER MEAN LENGTHS.
       I=NPAR + K
      FLL = X(NPAR+1)
```

```
      FLU= X(NPAR+K)
C  ADD THE PENALTY FUNCTION
      DO 145 J=1,K
      JJ=NPAR + J
      IF ( X(JJ) .LT.  FMMIN(J)) L=L-Q9*10.*( X(JJ)-FMMIN(J) )**4
      IF ( X(JJ) .LT. FMMIN(J)) G(JJ)=G(JJ)-40.*Q9*
     *   (X (JJ) - FMMIN(J))**3
      IF ( X(JJ) .GT. FMMAX(J) ) L=L-Q9*10.*(X(JJ)-FMMAX(J))**4
      IF ( X(JJ) .GT. FMMAX(J)) G(JJ)= G(JJ)- 40.*Q9*(X(JJ)-FMMAX(J))
     *   **3
  145 CONTINUE
  146 CONTINUE
  370 CONTINUE
C
C  ***************************************************************************
C CHAIN RULE CALCULATIONS FOR THE GRADIENT
C  ***************************************************************************
C
C  CALCULATE THE GRADIENT IN TERMS OF THE INTERNAL SUBROUTINE
C  VARIABLES
C
      DO 400 J=1,K
      TEMP = 2.* PP* ( X(NPAR+J) - MM(J)  )
      G(1)=G(1)+TEMP
      G(2)=G(2)+TEMP*USUM(J)/USUM(K)
      IF (J .NE. K) G(3)=G(3)+TEMP*X(2)*(DUSUM(J)-USUM(J)*DUSUM(K)/
     1   USUM(K))/USUM(K)
      IF ( II5 .EQ. 1 ) GO TO  400
      G(4)=G(4)+L2(J)*T5(J)
      IF ( II2 .EQ. 1 ) GO TO 400
      G(5)=G(5)+L2(J)*T6(J)
  400 CONTINUE
  410 CONTINUE
C
C  CALCULATE THE GRADIENT IN TERMS OF THE X VARIABLES FOR DFMIN
C
      DO 68 J=1,K
      DVAR= (U6/FMD) /(2.*T(J) *FMD)
      G(NPAR+J) = G(NPAR+J) + L1(J) - 2. * PP*( M(J) -MM(J) )
   68 CONTINUE
C
      ITEMP=NPAR+K
        DO 72 J=1,K
        G(J+ITEMP)=L3(J)*P(J)
C
          DO 70 J1=1,K
          G(J+ITEMP)=G(J+ITEMP)-L3(J1)*P(J1)*P(J)
   70     CONTINUE
C
   72   CONTINUE
C
        DO 73 J=1,K
        G(J+ITEMP)=G(J+ITEMP)-200.*ALOG(PSUM)*P(J)
   73   CONTINUE
C
```

```
      G(3)=-2.0*G(3)*X(3)*UINV**2
      G(4) =  G(4) * 2. * X(4)
      IF ( VFLAG .LT. 0.D0 ) G(5) = - G(5) * 2. * X(5)
      IF ( VFLAG .GT. 0.D0 ) G(5) =   G(5) * 2. * X(5)
      IF ( II8 .EQ. 1  ) G(6) = 0.D0
   78 CONTINUE
C
C  CALCULATE INITIAL PERCENTAGE ESTIMATES WITH FIXED MEANS AND
C  OTHER PARAMETERS.
C
      IF ( (  IIO .EQ.2 ) .OR. ( II11 .EQ. 2 )  ) GO TO 79
C
      DO 82 I=1,NPAR + K
      G(I) = 0.
   82 CONTINUE
   79 CONTINUE
C
C MULTIPLY FUNCTION BY -2
C
      DO  80   J= 1, N
      G(J) = -G(J) * 2.
   80 CONTINUE
      F =- L *2.
      RETURN
      END
C
C ******************
C  SUBROUTINE ODFMIN
C ******************
      SUBROUTINEODFMIN(F,X,G,N,W,DFN,MAXFN,IPRINT,EPS,IEXIT, IMAX,IMODE,
     *  H,MODE,
     *             N1,IJ,JK,DF,ICON,NN,I,IK, ICOUNT, FY, IS,
     *   J, II, GS, GYS, LLOG, IU, Z, K, GSO, DGS, IFN, IV, I1, DMIN,
     *   ALPHA,LINK,NP,IB,ZZ,ITN,TOT,SIG,IRETURN,IHANG)
C
C     ****************************************************************
C  VERSION OF DFMIN SUITABLE FOR SEGMENTATION
C     ****************************************************************
C
      DIMENSION X(1),G(1),H(1),W(1),EPS(1),FUNVAL(10),XSAVE(40)
      INTEGER TITLE (20)
      COMMON /SWBLK/  TITLE,
     .IIO,II1,II2,II3,II4,II5,II6,II7,II8,II9,II10,II11,
     .II12,II13,II14,II15
      LOGICAL LLOG
      IF ( IRETURN .EQ.  1  ) GO TO 8420
      IF ( IRETURN .EQ.  2  ) GO TO 8421
    2 CONTINUE
C
        DO 810 I=1,10
        FUNVAL(I)=0.0
  810   CONTINUE
C
      LLOG=.TRUE.
      NP=N+1
```

```
      N1=N-1
      NN=N*NP/2
      IS=N
      IU=N
      IV=N+N
      IB=IV+N
      IEXIT=0
      IF(MODE.EQ.3)GOTO15
      IF(MODE.EQ.2)GOTO10
      IJ=NN+1
      DO 5 I=1,N
      DO 6 J=1,I
      IJ=IJ-1
6     H(IJ)=0.D0
5     H(IJ)=1.D0
      GOTO15
10    CONTINUE
      IJ=1
      DO 11 I=2,N
      Z=H(IJ)
      IF(Z.LE.0.D0) GO TO 7020
      IJ=IJ+1
      II=IJ
      DO 11 J=I,N
      ZZ=H(IJ)
      H(IJ)=H(IJ)/Z
      JK=IJ
      IK=II
      DO 12 K=I,J
      JK=JK+NP-K
      H(JK)=H(JK)-H(IK)*ZZ
12    IK=IK+1
11    IJ=IJ+1
      IF(H(IJ).LE.0.D0)GO TO 7020
15    CONTINUE
      IJ=NP
      DMIN=H(1)
      DO 16 I=2,N
      IF(H(IJ).GE.DMIN)GOTO16
      DMIN=H(IJ)
16    IJ=IJ+NP-I
      IF(DMIN.LE.0.D0)GO TO 7020
      IF(DFN.EQ.0.D0) Z=F
      ITN=0
      IRETURN = 1
      RETURN
 8420 CONTINUE
C
        DO 4500 I5=1,N
        XSAVE(I5)=X(I5)
 4500   CONTINUE
C
      FUNVAL(10)=F
      IFN = IFN + 1
      DF=DFN
```

```
      IF(DFN.EQ.0.D0)DF=F-Z
      IF(DFN.LT.0.D0)DF=ABS(DF*F)
      IF(DF.LE.0.D0)DF=1.
 20   CONTINUE
 C
        DO 840 I=1,9
        FUNVAL(I)=FUNVAL(I+1)
  840   CONTINUE
      FUNVAL(10)=F
 C
      IF (ITN .LE. 15 .OR. ABS(FUNVAL(1)-FUNVAL(10)) .GT. .001) GO TO
     1 846
      IHANG=1
      RETURN
 C
  846 ICONV=1
        DO 1021 I=1,N
        IF (ABS(G(I)) .GT. EPS(I)) ICONV=2
 1021   CONTINUE
 C
      IF (ICONV .NE. 1) GO TO 1022
      IEXIT=1
      GO TO 92
 C
 1022 IF (IPRINT .EQ. 0) GO TO 21
      IF(ITN.EQ.0) GO TO 7000
      IF(MOD(ITN,IPRINT).NE.0) GO TO 21
      IF(LLOG) GO TO 7010
 7003 WRITE(109,1001) ITN,IFN,F
 1001 FORMAT(/1H , 'ITER. NO.  ', I5, '   FCN EVALUATIONS  ',
     *I5,'  FCN VALUE:',F20.7)
 1102 FORMAT(1H ,'CO-ORDINATES'/((5F16.7)))
 1103  FORMAT(1H ,'GRADIENT'/ ((5F16.7)))
 1105 FORMAT(' FUNCTION VALUE= ',F15.7)
 7002 IF(IPRINT.LT.0) GO TO 21
      WRITE(109,1102) (X(I),I=1,N)
      WRITE(109,1103) (G(I),I=1,N)
 21   CONTINUE
      DO 4135 I5=1,N
         XSAVE(I5)=X(I5)
 4135 CONTINUE
      ICOUNT = 0
      ITN=ITN+1
      W(1)=-G(1)
      DO 22 I=2,N
      IJ=I
      I1=I-1
      Z=-G(I)
      DO 23 J=1,I1
      Z=Z-H(IJ)*W(J)
 23   IJ=IJ+N-J
 22   W(I)=Z
      W(IS+N)=W(N)/H(NN)
      IJ=NN
      DO 25 I=1,N1
```

```
      IJ=IJ-1
      Z=0.DO
      DO 26 J=1,I
      Z=Z+H(IJ)*W(IS+NP-J)
26    IJ=IJ-1
25    W(IS+N-I)=W(N-I)/H(IJ)-Z
      GS=0.DO
      DO 29 I=1,N
29    GS=GS+W(IS+I)*G(I)
      IEXIT=2
      IF(GS.GE.0.DO)GOTO92
      GSO=GS
      ALPHA=-2.DO*DF/GS
      IF(ALPHA.GT.1.DO)ALPHA=1.DO
      DF=F
      TOT=0.DO
30    CONTINUE
      IEXIT=3
      IF(IFN.EQ.MAXFN)GOTO92
      IEXIT=1
      DO 31 I=1,N
      Z=ALPHA*W(IS+I)
31    X(I)=X(I)+Z
      IRETURN = 2
      RETURN
 8421 CONTINUE
      ICOUNT = ICOUNT + 1
      IFN=IFN+1
      IF (ICOUNT .LT. IMAX .OR. IMODE .NE. 2) GO TO 33
C
        DO 4600 I5=1,N
        X(I5)=XSAVE(I5)
 4600   CONTINUE
C
      IRETURN=4
      IEXIT=1
      GO TO 92
C
   33 GYS=0.0
      DO 32 I=1,N
32    GYS=GYS+W(I)*W(IS+I)
      IF(FY.GE.F)GOTO40
      IF(ABS(GYS/GSO).LE..9DO)GOTO50
      IF(GYS.GT.0.DO)GOTO40
      TOT=TOT+ALPHA
      Z=10.DO
      IF(GS.LT.GYS)Z=GYS/(GS-GYS)
      IF(Z.GT.10.DO)Z=10.DO
      ALPHA=ALPHA*Z
      F=FY
      GS=GYS
      GOTO30
40    CONTINUE
      DO 41 I=1,N
41    X(I)=X(I)-ALPHA*W(IS+I)
```

```
      Z=3.DO*(F-FY)/ALPHA+GYS+GS
      ZZ=SQRT(Z*Z-GS*GYS)
      Z=1.DO-(GYS+ZZ-Z)/(2.DO*ZZ+GYS-GS)
      ALPHA=ALPHA*Z
      GOTO30
50    CONTINUE
      ALPHA=TOT+ALPHA
      F=FY
      DF=DF-F
      DGS=GYS-GSO
      LINK=1
      IF(DGS+ALPHA*GSO.GT.0.DO)GOTO52
      DO 51 I=1,N
51    W(IU+I)=W(I)-G(I)
      SIG=1.DO/(ALPHA*DGS)
      GOTO70
52    CONTINUE
      ZZ=ALPHA/(DGS-ALPHA*GSO)
      Z=DGS*ZZ-1.DO
      DO 53 I=1,N
53    W(IU+I)=Z*G(I)+W(I)
      SIG=1.DO/(ZZ*DGS*DGS)
      GOTO70
60    CONTINUE
      LINK=2
      DO 61 I=1,N
61    W(IU+I)=G(I)
      IF(DGS+ALPHA*GSO.GT.0.DO)GOTO62
      SIG=1.DO/GSO
      GOTO70
62    CONTINUE
      SIG=-ZZ
      GOTO70
65    CONTINUE
      DO 66 I=1,N
66    G(I)=W(I)
      GOTO20
70    CONTINUE
      W(IV+1)=W(IU+1)
      DO 71 I=2,N
      IJ=I
      I1=I-1
      Z=W(IU+I)
      DO 72 J=1,I1
      Z=Z-H(IJ)*W(IV+J)
72    IJ=IJ+N-J
71    W(IV+I)=Z
      IJ=1
      DO 75 I=1,N
      Z=H(IJ)+SIG*W(IV+I)*W(IV+I)
      IF(Z.LE.0.DO)Z=DMIN
      IF(Z.LT.DMIN)DMIN=Z
      H(IJ)=Z
      W(IB+I)=W(IV+I)*SIG/Z
      SIG=SIG-W(IB+I)*W(IB+I)*Z
```

```
75     IJ=IJ+NP-I
       IJ=1
       DO 80 I=1,N1
       IJ=IJ+1
       I1=I+1
       DO 80 J=I1,N
       W(IU+J)=W(IU+J)-H(IJ)*W(IV+I)
       H(IJ)=H(IJ)+W(IB+I)*W(IU+J)
80     IJ=IJ+1
       GOTO(60,65),LINK
       DO 91 I=1,N
91     G(I)=W(I)
92     CONTINUE
       IF(IEXIT.EQ.2) WRITE(108,7023)
7023   FORMAT('-*** GRAD TRANSPOSE TIMES DELTA X GREATER THAN OR ',
     * 'EQUAL ZERO'/6X, '  EPS SET TOO SMALL?')
       IF(IEXIT.EQ.3) WRITE(108,7092)
7092   FORMAT('-*** MAXIMUM NUMBER OF FUNCTION EVALUATIONS EXCEEDED')
       IF(IPRINT.EQ.0) GO TO 777
       WRITE(109,1006)
       WRITE(109,1004) ITN,IFN,IEXIT
1006   FORMAT('-FINAL STATISTICS :'/'+_________________'/)
 1004  FORMAT(/1H , '   ITERATION NO.   ',I5, '     FUNCTION EVALUATIONS
     *   ',I5, '     EXIT CODE  ',I2)
       WRITE(109,1005) F
       WRITE(109,1002) (X(I),I=1,N)
       WRITE(109,1003) (G(I),I=1,N)
1002   FORMAT(1H ,'CO-ORDINATES'/(1X,(6D20.12)))
1005   FORMAT(' FUNCTION VALUE= ',D20.12)
1003    FORMAT(1H ,'GRADIENT',/(1X,(6D20.12)))
 777   CONTINUE
       IRETURN=4
       IF (IEXIT .NE. 1 ) IRETURN =3
C 777    IF(IEXIT.NE.1) RETURN
       RETURN
C *** PRINT HEADINGS **
7000   WRITE(109,7001)
7001   FORMAT('1INITIAL STATISTICS :'/'+_________________'/)
       GO TO 7003
7010   WRITE(109,7011)
7011   FORMAT('-INTERMEDIATE STATISTICS :'/'+_____________________')
       LLOG=.FALSE.
       GO TO 7003
7020   WRITE(108,7021)
7021   FORMAT('-*** HESSIAN NOT POSITIVE DEFINITE')
       IRETURN = 3
       RETURN
       END
C
C*******************
C SUBROUTINE GETINPUT
C*******************
       SUBROUTINE   GETINPUT( N,X,NPAR,FREQ,PMIN,PMAX,IMODE,MAXFN,IPRINT,
     *  IMAX,N5,W5,S5,S6, FLMIN, FLMAX, FHMIN, FHMAX, FKMIN,
     *   PP, SDMIN, SDMIN1, FTOT, TITLE  ,V1MAX, VKMAX  ,VKMIN, FMMIN,
```

```
     *    FMMAX , OMEGA , CC, BBETA ,BBETA1, VFLAG ,PR12M, PRK2M, PRKLM,
     *    PRKUM ,PR,JMIN,JMAX,FREQTO,NTRY,II17,FMORT,NFULL,
     *    PENMOR )
C
C     ******************************************************************
C  THIS SUBROUTINE READS IN THE INITIAL USERS ESTIMATES, CALCULATES
C  THE INITIAL POPULATION ESTIMATES ( THROUGH THE SUBROUTINE POPEST )
C  AND TRANSFORMS THE VARIABLES TO THE FORM USED BY THE OTHER
C  PARTS OF THE PROGRAM.  IT CHECKS THAT INITIAL ESTIMATES ARE WITHIN
C  BOUNDS, AND WRITES ERROR MESSAGES IF THEY ARE NOT.
C     ******************************************************************
C
      INTEGER TITLE(40)
      DIMENSION FREQ(1), X(1), PMIN(1), PMAX(1)
      DIMENSION FMMIN(1), FMMAX(1)
      INTEGER W5
      DIMENSION CC(20),OMEGA(20),WW(20)
      DIMENSION BBETA1(1)
      DIMENSION JMIN(1), JMAX(1)
      DIMENSION PR(1,1)
      DIMENSION PTEMP(30)
      ITROLL=0
C    ITROLL IS AN ERROR FLAG.  IF 1, THE PROGRAM STOPS BELOW.
C
C    READ # AGE CLASSES, MAXIMUM FUNCTION EVALS, AND
C    THE AGE-CLASS# OF THE FIRST FULLY RECRUITED AGE-CLASS
      READ(105,*) NK, MAXFN, NFULL
      IF (NFULL.GT.NK) THEN
        WRITE (108,1201)
1201    FORMAT ('  ERROR: NFULL IS GREATER THAN NK')
        ITROLL=1
      END IF
C
C  (WITHIN THIS PROGRAM, NK IS CALLED K.
C  IF YOU'RE TRYING TO DEBUG THE MODEL, DON'T LET THIS FOUL YOU UP)
      K=NK
      NTRY=0
      II17=0
      IMODE=2
C    READ ESTIMATED MORTALITY AND ITS ASST'D PENALTY
      IF ( NFULL .NE. 0 ) READ(105,*) FMORT,PENMOR
C
      READ(105,*) IPRINT
      IMAX=30
C
C     READ THE #FREQUENCIES, THE FIRST AGE, THE LOWER
C     LIMIT OF THE FIRST LENGTH INTERVAL, AND THE
C     INTERVAL WIDTH IN USER UNITS.
      READ(105,*) N5, W5, S5, S6
C
C    READ THE FREQUENCIES
      READ(105,*) ( FREQ(I), I=1,N5)
C
C    READ THE LOWER BOUND ON VBK, ITS ASST'D PENALTY,
C     THE LOWER BOUND ON THE S.D. OF THE FIRST AGE CLASS,
```

```
C     THE UPPER BOUND ON THE S.D. OF THE FIRST AGE CLASS,
C     THE LOWER BOUND ON THE S.D. OF THE LAST AGE CLASS, &
C     THE UPPER BOUND ON THE S.D. OF THE LAST AGE CLASS.
      READ(105,*) FKMIN, PP, SDMIN, V1MAX, VKMIN, VKMAX
      PP = 1.0/PP
      BBETA = 0.0
      FMULT = 1.0
      VFLAG = 0.0
C
C    READ THE LOWER BOUNDS ON THE POP'N PROPORTIONS
      READ(105,*) ( PMIN(I), I=1,K)
C    READ THE UPPER BOUNDS ON THE POP'N PROPORTIONS
      READ(105,*) ( PMAX(I), I=1,K)
C

C    READ THE LOWER BOUNDS ON THE MEAN LENGTHS
      READ(105,*) ( FMMIN(I), I=1,K)
C    READ THE UPPER BOUNDS ON THE MEAN LENGTHS
      READ(105,*) ( FMMAX(I), I=1,K)
C
C    READ THE ESTIMATED MEAN LENGTH FOR THE FIRST AGE CLASS,
C       THE ESTIMATED MEAN LENGTH FOR THE LAST AGE CLASS,
C       AN ESTIMATE FOR VBK, AN ESTIMATE FOR S.D. OF THE
C       FIRST AGE CLASS, AND THE DIFFERENCE IN S.D. BETWEEN
C       THE FIRST AND LAST AGE CLASSES.
      READ (105,*)  (X(I), I=1,5)
      X(6)=0.0
C    CHECK MEANS AGAINST BOUNDS
      IF (X(1).LT.FMMIN(1).OR.X(1).GT.FMMAX(1)) THEN
        WRITE (108,1202)
1202    FORMAT ('  ERROR: FIRST OR LAST MEAN OUTSIDE THE BOUNDS')
        ITROLL = 1
      END IF
      IF (X(2).LT.FMMIN(K).OR.X(2).GT.FMMAX(K)) THEN
        WRITE (108,1202)
        ITROLL = 1
      END IF
C
C    CHECK ESTIMATED BRODY COEFFICIENT
      IF (X(3).LT.FKMIN) THEN
        WRITE(108,1203)
1203    FORMAT ('  ERROR: INITIAL BRODY COEFFICIENT BENEATH BOUND')
        ITROLL = 1
      END IF
C
C   CHECK FIRST S.D.
      IF (X(4).LT.SDMIN.OR.X(4).GT.V1MAX) THEN
        WRITE (108,1204)
1204    FORMAT ('  ERROR: FIRST STANDARD DEVIATION OUTSIDE BOUNDS')
        ITROLL = 1
      END IF
    2 N=2*K+NPAR
      IF ( FMULT .EQ. 0.D0 ) FMULT = 1.
      IF ( V1MAX .EQ. 0.D0) V1MAX = 10000.
      IF ( VKMAX .EQ. 0.D0) VKMAX = 10000.
```

```
      UMIN = FKMIN
      IF ( VFLAG .NE. 0.D0 ) X(5) = X(5) + 1.D-2
      FTOT = 0.
      DO 826 I=1,    N5
      FTOT = FTOT + FREQ(I)
  826 CONTINUE
C
C     GET INITIAL POPULATION ESTIMATES
C
      CALL POPEST(K,X,FREQ,N5,S5,S6,NPAR,FTOT,PLAST,ITROLL)
C        (POPEST READS THE ESTIMATED POP'N PROPORTIONS)
C    CHECK PROPORTIONS AGAINST BOUNDS
        DO 1210 I=1,K
          IF (X(NPAR+K+I).LT.PMIN(I).OR.X(NPAR+K+I).GT.PMAX(I)) THEN
          WRITE (108,1211) I
1211      FORMAT(' ERROR: POPULATION PROPORTION ',I2,' OUT OF BOUNDS')
          TROLL = 1
          END IF
1210    CONTINUE
C
        DO 1400 J=1,K
        X(NPAR+K+J)=ALOG(X(NPAR+K+J))
 1400   CONTINUE
C
      ITEMP = NPAR+K
      FLMAX=FMMAX(1)
      FLMIN= FMMIN(1)
      FHMIN= FMMIN(K)
      FHMAX=FMMAX(K)
C
      IF ( NFULL .NE. 0 ) WRITE(108,5640) NFULL,FMORT,PENMOR
 5640 FORMAT(/'   YOUNGEST FULLY RECRUITED AGE CLASS IS ',I5,/,
     *   ' INSTANTANEOUS NAT MORT IS ',F3.2 ,' PENALTY IS',F4.1/)
C
C  COMPUTE THE VALUES FOR OMEGA AND CC
C
      DO 1506 J=1,K
      BBETA1(J)= BBETA*(FLOAT(K-J+1)/FLOAT(K))** FMULT
      CC(J)=0.5
      OMEGA(J)=1/SQRT(2*3.14159)
 1506 CONTINUE
      WRITE(108,12) K,N, MAXFN
   12 FORMAT( '  NUMBER OF AGE CLASSES',I4,'   NUMBER OF VARIABLES',I4,
     *  /,'   MAXIMUM NUMBER OF FUNCTION EVALUATIONS
     *' ,I5     )
      WRITE(108,20) IPRINT
   20 FORMAT(    '0    THE NUMBER OF ITERATIONS PER PRINTOUT IS', I4)
      WRITE(108,40)N5, W5, S5, S6
   40 FORMAT('0   THE NUMBER OF LENGTH FREQUENCIES IS',I4,
     *  '   YOUNGEST AGE CLASS', I4,/,'   SHORTEST LENGTH',F7.2,
     *    '   INTERVAL LENGTH',F6.2)
      WRITE(108,70)
   70 FORMAT(/,'-   THE LENGTH FREQUENCIES ARE',/)
      WRITE(108,52) ( FREQ(I),I=1,N5)
   52 FORMAT(10F6.1)
```

```
      WRITE(108,60) UMIN,PP,SDMIN,V1MAX,VKMIN,VKMAX
   60 FORMAT(1X,'LOWER BOUND ON BRODY COEFF.: ',F6.3,/,' VARIANCE OF
     1 MEANS AROUND VON BERTALANFFY: ',F5.2,/,1X,'BOUNDS ON FIRST AND '
     2 ,'LAST STANDARD DEVIATION: ',1X,4F7.2)
      WRITE(108,80)
   80 FORMAT(/,' LOWER AND UPPER BOUNDS ON THE PROPORTIONS ARE:',/)
      WRITE(108,50) (PMIN(I) ,I=1,K)
      WRITE(108,50) (PMAX(I) ,I=1,K)
      WRITE(108,62)
   62 FORMAT(/,' LOWER AND UPPER BOUNDS ON THE MEAN LENGTHS ARE:',/)
      WRITE(108,50) ( FMMIN(I),I=1,K)
      WRITE(108,50) ( FMMAX(I),I=1,K)
      WRITE(108,827)
  827 FORMAT(/,' INITIAL X VALUES ARE:',/)
      WRITE(108,50) (X(I), I=1,NPAR)
C
C  TRANSFORM FROM THE "LLOWER, LUPPER, K" PARAMETRIZATION TO THE
C  "V, C, K" PARAMETRIZATION
C
      TEMP1= ( X(2) - X(1) ) * EXP( X(3) ) **W5
     *     /( 1. - EXP( X(3) ) ** ( -K+1)  )
      TEMP2=  ( X(2) - X(1) * EXP( X(3) )**(-K+1) )
     * / (( X(2) - X(1) ) *   EXP(X(3)) ** W5 )
      TEMP3 = SQRT( EXP( X(3) ) - EXP ( FKMIN )  )
      X(2)=X(2)-X(1)
      X(3) = TEMP3
      UMIN = EXP( FKMIN )
      U=UMIN+X(3)**2
      UINV=1.0/U
C
C SET UP INITIAL STANDARD DEVIATION
C
      X(4)=SQRT(X(4)-SDMIN)
C
C  CALCULATE THE INITIAL MEANS
C
        DO 200 J=1,K
        X(NPAR+J)=X(1)+X(2)*(1.0-UINV**(J-1))/(1.0-UINV**(K-1))
C    CHECK MEANS AGAINST BOUNDS
        IF (X(NPAR+J).LT.FMMIN(J).OR.
     *   X(NPAR+J).GT.FMMAX(J)) THEN
         WRITE (108,1205) J
1205     FORMAT ('  ERROR: MEAN',I2,' OUT OF BOUNDS')
         ITROLL = 1
        END IF
  200   CONTINUE
C
      PP= 1./ PP
      WRITE(108,160)
      WRITE(108,50)  (X(I), I= NPAR+1,NPAR+K )
  160 FORMAT(/,' THE INITIAL LENGTHS ARE:',/)
  180 FORMAT(/,' THE INITIAL ESTIMATES FOR THE POPULATION',
     *' PROPORTIONS ARE:',/)
      WRITE(108,180)
C
```

```
        DO 190 J=1,K
        WW(J)=EXP(X(ITEMP+J))
  190   CONTINUE
C
      WRITE(108,50) (WW(J),J=1,K)
   50 FORMAT(10F7.2)
      READ(105,175) (TITLE(I),I=1,40)
175   FORMAT (40A2)
      WRITE(108,1501)FTOT
 1501 FORMAT(' THE TOTAL NUMBER OF FISH IS',F6.0)
C
 1668 CONTINUE
   30 FORMAT(I3,I2,2F4.0)
C
   54 FORMAT(F4.2,F4.1,4F5.1,3F4.1)
C
      IF (ITROLL.Eq.1) GO TO 1299
      RETURN
1299  CONTINUE
      CALL EXIT
      END
C
C
C ******************
C  SUBROUTINE EPSSET
C ******************
      SUBROUTINE EPSSET(N,EPS,IPHASE,K)
C
C
      DIMENSION EPS(1)
      DATA LP/108/
C
      IF (IPHASE .NE. 0) GO TO 200
C
C ONLY POPULATION PERCENTAGES
C
        DO 100 I=1,N
        EPS(I)=.5
  100   CONTINUE
      RETURN
C
  200 IF (IPHASE .NE. 1) GO TO 400
C
C FIXED VARIANCE
C
        DO 300 I=1,N
        EPS(I)=10.
  300   CONTINUE
      RETURN
C
  400 IF (IPHASE .NE. 2) GO TO 700
C
C CONSTANT VARIANCE
C
        DO 500 I=1,3
```

```
      EPS(I)=5.
  500 CONTINUE
C
      DO 600 I=4,N
      EPS(I)=.5
  600 CONTINUE
      RETURN
C
  700 IF (IPHASE .NE. 3) GO TO 1000
C
C LINEAR VARIANCE
C
      DO 800 I=1,6
      EPS(I)=.1
  800 CONTINUE
C
      DO 900 I=7,N
      EPS(I)=.01
  900 CONTINUE
      RETURN
C
 1000 WRITE(108,1010) IPHASE
 1010 FORMAT(' ',/   ,5X,'*** ERROR ***  IPHASE OUT OF RANGE: IPHASE=',
     1 I5)
      CALL EXIT
      END
C
C ***************
C SUBROUTINE SSET
C ***************
      SUBROUTINE SSET(IPHASE,II0,II2,II5,II8,II11)
C
      DATA LP/108/
C
      IF (IPHASE .NE. 0) GO TO 100
      II0=1
      II2=2
      II5=2
      II8=2
      II11=1
      RETURN
C
  100 IF (IPHASE .NE. 1) GO TO 200
      II0=2
      II2=2
      II5=1
      II8=2
      II11=2
      RETURN
C
  200 IF (IPHASE .NE. 2) GO TO 300
      II0=2
      II2=1
      II5=2
      II8=2
```

```
      III1=2
      RETURN
C
  300 IF (IPHASE .NE. 3) GO TO 400
      II0=2
      II2=2
      II5=2
      II8=1
      III1=2
      RETURN
C
  400 IF (IPHASE .NE. 4) GO TO 500
      WRITE(108,410)
  410 FORMAT(' ',/   ,5X,'ALL PHASES COMPLETED')
C
      CALL EXIT
C
  500 WRITE(108,510) IPHASE
  510 FORMAT(' ',/   ,5X,'*** ERROR ***  IPHASE OUT OF RANGE: IPHASE=',
     1 I5)
      CALL EXIT
      END
C
C ******************
C  SUBROUTINE POPEST
C ******************
      SUBROUTINE POPEST(K,X,FREQ,N5,S5,S6,NPAR,FTOT,PLAST,ITROLL)
      DIMENSION X(1), FREQ(1), AV(25), P(25)
      ITEMP = NPAR + K
      PLAST = 1.DO
C   READ THE ESTIMATED POP'N PROPORTIONS.
      READ(105,*) ( X(J),J=ITEMP+1,ITEMP+K-1)
C
        DO 385 J=1,K-1
        PLAST = PLAST -   X(ITEMP+J)
385     CONTINUE
      X(ITEMP+K)=PLAST
C  CHECK PROPORTIONS ADD UP TO LESS THAN 1.DO
      IF (PLAST.LT.0.DO) THEN
         WRITE (108,1201)
1201     FORMAT ('  ERROR: PROPORTIONS SUM TO MORE THAN 1.0')
         ITROLL = 1
      END IF
C
      RETURN
      END
C
C ******************
C  SUBROUTINE FINPRT
C ******************
      SUBROUTINE FINPRT(K,X,F,G,NPAR,TITLE, SUMIN, SUMIN1,FKMIN,
     *   NFULL,PENMOR,UMIN ,W5,FLMIN,FLMAX,FHMIN,FHMAX,PP)
C
C  THIS SUBROUTINE PRINTS OUT THE FINAL ESTIMATES.
C
```

```
      INTEGER TITLE(40)
      DIMENSION X(1), G(1)
      DIMENSION  SD(25)
      REAL MM(25)
      INTEGER W5
      DIMENSION IMM(16)
      WRITE(108,100)(TITLE(I), I=1,40 )
  100 FORMAT( '1 ', 40A2,/)
C
C  CALCULATE VON BERTALANFFY PARAMETERS
C
      UINV=1.0/(UMIN+X(3)**2)
      XX=X(1)+X(2)/(1.0-UINV**(K-1))
      YY = ALOG(X(3)**2+ UMIN)
      ZZ=W5+ALOG(X(2)/(1.0-UINV**(K-1))/XX)/YY
      WRITE(108,200) XX,YY,ZZ
  200 FORMAT(' THE VON BERTALANFFY PARAMETERS:',/,' LINFINITY='
     1,F7.2,5X, 'K=',F7.3,5X,'TZERO=',F7.3,/)
      WRITE(108,300)
  300 FORMAT(' THE ESTIMATES FOR THE MEAN LENGTHS ARE:'  )
      WRITE (108,400)  ( X(J) , J= NPAR+1, NPAR+K  )
  400 FORMAT(   10F7.2)
      ITEMP = NPAR+K
      PLAST = 1.
      PSUM=0.
        DO 600 J=1,K
        MM(J)=EXP(X(ITEMP+J))
        PSUM=PSUM+MM(J)
  600   CONTINUE
C
        DO 610 J=1,K
        MM(J)=MM(J)/PSUM
  610   CONTINUE
C
      IF ( PLAST .LT. 0. ) PLAST = 0.
      WRITE(108,500)
  500 FORMAT(/,' ESTIMATES FOR THE POPULATION PROPORTIONS ARE:
     *'  )
      WRITE(108,401) (MM(J),J=1,K)
  401 FORMAT (10F7.3)
C
      IF ( NFULL .EQ. 0 ) GO TO 8434
      TEMP=0.
      PBAR=0.
      DO 8432 J=NFULL,K
      PBAR=PBAR+ALOG(MM(J))
 8432 CONTINUE
      PBAR=PBAR/FLOAT(K-NFULL+1)
      RBAR= FLOAT(K-NFULL)/2.
      FMORT=0.
      DO 8433 J=NFULL,K
      FMORT=FMORT + ( ALOG(MM(J))-PBAR)*(J-NFULL-RBAR)
      TEMP=TEMP+(J-NFULL-RBAR)**2
 8433 CONTINUE
      FMORT=-FMORT/TEMP
```

```
      WRITE(108,8437) FMORT
 8437 FORMAT(/' THE ESTIMATED TOTAL MORTALITY RATE IS',F8.3)
 8434 CONTINUE
C
      TEMP = SDMIN+ X(4)**2
      FM = ( FLMIN + FLMAX )/2.
      FMU=( FHMIN + FHMAX)/2.
      FMD = FMU-FM
      DO 750 J=1,K
      TEMP = FLOAT( J-1)/ FLOAT(K-1)
      TEMP=SQRT(TEMP)
      SD(J) = SDMIN + X(4)**2 + X(5) * TEMP + X(6) * TEMP *( TEMP-1.)
  750 CONTINUE
      WRITE(108,800)
  800 FORMAT(/,' THE ESTIMATED STANDARD DEVIATIONS OF THE LENGTHS',
     *' AROUND THE MEAN LENGTHS ARE: ')
      WRITE(108,400) ( SD(J), J=1,K )
      TEMP = 0.
      U= X(3)**2 + UMIN
      UINV=1.0/U
C
C  CALCULATE ESTIMATE OF THE VARIANCE OF THE MEAN LENGTHS AROUND THE
C  VON BERTALANFFY CURVE
C
        DO 920 J=1,K
        MM(J)=X(1)+X(2)*(1.0-UINV**(J-1))/(1.0-UINV**(K-1))
  920   CONTINUE
C
      DO 950 J=1,K
      TEMP = TEMP+ ( X( NPAR+J)- MM(J) ) **2
  950 CONTINUE
      TEMP = TEMP / K
      WRITE (108,1100) TEMP
 1100 FORMAT(/,' ESTIMATED VARIANCE OF MEAN LENGTHS AROUND',/,
     *' THE VON BERTALANFFY CURVE IS', F9.5,/)
      WRITE(108,1200) F
 1200 FORMAT( ' THE OBJECTIVE FUNCTION IS  ', G12.5  )
C
      RETURN
      END
```

APPENDIX 2. Fortran source code for the length frequency simulator described in this report.

```
C*************************************************************************
C  SIMULATOR FOR GENERATING OBSERVED LENGTH FREQUENCIES OF
C  FISH WITH VONBERTELANFY GROWTH CHARACTERISTICS
*************************************************************************
C
      REAL M(30),L1,L2,LINT,L
      REAL*8 IX0,IX1
      REAL*8 SEED
      DIMENSION PASIM(30)
      COMMON /OVBLK/          NFREQ,    MINAG,    L1,       LINT,
     *   F(200),  INDEX,     JMIN(100), JMAX(100),FREQA(120),RF(100),
     .   M,         S(30),     P(30),    PSIM(30),  L,       MAXAG,
     .   FAL(100,10),NAGE,    PSUM,     RTOT,     SEED,
     *   SFAL(100,10),INDEXS,SFREQA(25),JSMIN(100),JSMAX(100),
     *   IARR(20),   ZMORT,   RSIG,   FMDEV(20),ITITLE(40),FMORT,
     *   ISEEDX ,  VBK
C
      CALL NASIM(PASIM)
      CALL LPOUT(PASIM)
      CALL EXIT
      END
C
      SUBROUTINE NASIM(PASIM)
      REAL*8 SEED
      REAL M(30), L1,L2,LINT,L
      DIMENSION KSWICH(16)
      DIMENSION PASIM(30)
      REAL*8 IX0,IX1,IX0A,IX1A
      DIMENSION PA(20,5)
      DIMENSION GAM(20)
      DIMENSION ISLEN(100), NSFISH(100)
            COMMON /OVBLK/          NFREQ,    MINAG,    L1,       LINT,
     *   F(200),  INDEX,     JMIN(100), JMAX(100),FREQA(120),RF(100),
     .   M,         S(30),     P(30),    PSIM(30),  L,       MAXAG,
     .   FAL(100,10),NAGE,    PSUM,     RTOT,     SEED,
     *   SFAL(100,10),INDEXS,SFREQA(25),JSMIN(100),JSMAX(100),
     *   IARR(20),   ZMORT,   RSIG,   FMDEV(20),ITITLE(40),FMORT,
     *   ISEEDX ,  VBK
      DATA IARR /1,9*2,3*0,1,6*0/
C
C  READ TITLE
      READ(10,8891) (ITITLE(I),I=1,40)
      WRITE(20,8892) (ITITLE(I),I=1,40)
8891  FORMAT(40A2)
8892  FORMAT(1X,40A2)
C   READ NUMBER OF OBSERVATIONS TO BE SIMULATED
      READ(10,*) NFISH
C   READ # OF AGE-CLASSES, FIRST AGE, AND # FREQUENCIES
      READ(10,*,END=9142)  NAGE,MINAG,NFREQ
C   READ LINFINITY, K, TZERO, AND THE S.D. PARAMETERS
      READ(10,* ) VBL, VBK, VBT, A, B
C   READ VARIANCE AROUND MEANS OF VB
```

```
      READ (10,*)VVB
      IARR(15) = IFIX (10*VVB)
C   READ THE S.D. RANGE AND THE SEED
      READ(10,*)    RANGE, ISEEDX
      IXO=ISEEDX
      II= IXO
      WRITE(20,7280)  IXO
 7280 FORMAT(/,'  THE SEED FOR RANDU IS ',F7.0 )
C   READ THE TOTAL MORTALITY RATE AND THE RECT S.D.
         READ(10,*) ZMORT,RSIG
         WRITE(20,76555)ZMORT,RSIG
76555    FORMAT(/'   TOTAL MORTALITY RATE IS ',F5.2,
     * /, '  STANDARD DEVIATION FOR RECRUITMENT IS ',F5.2)
C
         CALL GETPS(NAGE,IXO,ZMORT,RSIG,P)
C
         TEMP=0.
         PBAR=0.
         DO 8432 J=1,NAGE
            PBAR=PBAR+ALOG(P(J))
 8432    CONTINUE
         PBAR=PBAR/FLOAT(NAGE)
         RBAR= FLOAT(NAGE-1)/2.
         FMORT=0.
         DO 8433 J=1,NAGE
            FMORT=FMORT + ( ALOG(P(J))-PBAR)*(J-1-RBAR)
            TEMP=TEMP+(J-1-RBAR)**2
 8433    CONTINUE
         FMORT=-FMORT/TEMP
         WRITE(20,8437) FMORT
 8437    FORMAT(/'   THE ESTIMATED TOTAL MORTALITY RATE WITH',
     *     '  PERFECT AGEING IS',F8.3)
C
C  INITIALIZE ARRAYS
C
      DO 5 I = 1, NFREQ
      JMIN(I) = -1
      JMAX(I) = -1
      RF(I) = 0.
      DO 6 J = 1, NAGE
      FAL(I,J) = 0.
    6 CONTINUE
    5 CONTINUE
      DO 4800 I=1,NFREQ
      JSMIN(I)=-1
      JSMAX(I)=-1
      DO 4810 J=1,NAGE
 4810 CONTINUE
 4800 CONTINUE
C
C  SET UP ERROR PROBABILITIES FOR AGEING
C
C     CALL RANDU(IXO,IX1,X)
```

```
      X=GGUBFS(IX0)
C     IX0=IX1
C THE PARAMETER VECTOR USED IN A SEPARATE LIKLIHOOD FUNCTION
C CALCULATION CORRESPONDS TO THE QUANTITIES:
C VBL,VBT,SQRT(VBK),SQRT(A),SQRT(B), P(I) 1 LE I LE NAGE-1
      SK= SQRT(VBK)
      SA= SQRT(A)
      SB = SQRT(B)
   90 FORMAT(' P(',I2,')',T15,F16.3)
      WRITE(20,6490)   VBL
6490 FORMAT(//,' L INFINITY',T15,F16.3)
      WRITE(20,6590)    VBT
6590 FORMAT(' T ZERO ',T15,F16.3)
      WRITE(20,6690)    VBK
6690 FORMAT(' VB K',T15,F16.3)
      WRITE(20,6790)    A
6790 FORMAT(' A PARAMETER',T15,F16.3)
      WRITE(20,6890)    B
6890 FORMAT(' B PARAMETER',T15,F16.3)
C
C  CALCULATE THE MEANS AND VARIANCES
C
      CALL GETMS(M,S,VBL,VBK,NAGE,MINAG,A,B,VBT,IARR,IX0,FMDEV)
C
C THIS MODEL SETS THE MEANS ACCORDING TO VONBERTALANFFY GROWTH
C AND SETS THE VARIANCES TO BE LINEAR FUNCTIONS OF THE MEANS.
C
      L2= M(NAGE) +RANGE*S(NAGE)
      L1= M(1)-RANGE *S(1)
C THE SMALLEST LENGTH, LARGEST LENGTH, AND LENTH INTERVAL ARE:
      LINT= (L2-L1)/NFREQ
      DO 500 I=1,NAGE
        PSIM(I)= 0.
        PASIM(I)=0.
  500 CONTINUE
      DO 600 I=1,NFREQ
        F(I)= 0.
  600 CONTINUE
C
C  FIRST GENERATE FISH FOR LENGTHS AND THEN FOR LENGTH-AGE
      IF ( NFISH .EQ. 0 ) GO TO 1010
C
C  GENERATE AGE AND LENGTH OF FISH
C
      DO 1000I=1, NFISH
      CALL FGEN( P, NAGE, IAGE, M, S, IL, NFREQ , IX0, IX1,L1,L2
     *   ,  L, LINT,IARR )
C
C  FISH LIES OUTSIDE OF REGION OF INTEREST
C
      IF ( IL .EQ. -1 ) GO TO 995
C
```

```
      F(IL)=F(IL)+1.
  995 PSIM ( IAGE ) = PSIM ( IAGE ) + 1.
 1000 CONTINUE
 1010 CONTINUE
C
C  GENERATE FISH AGE-LENGTH AND APPARANT AGE
C
      PSUM = 0.
      PASUM=0.
      DO 1500 I= 1,NAGE
        PSUM = PSUM+PSIM(I)
        PASUM=PASUM+PASIM(I)
 1500 CONTINUE
      DO 1600 I=1,NAGE
        IF(PSUM.NE.0)PSIM(I)=PSIM(I)/PSUM
        IF (PASUM.NE.0)PASIM(I)=PASIM(I)/PASUM
 1600 CONTINUE
      INDEX = 0.
      DO 1610 I=1,NFREQ
      J1 = JMIN(I)
      J2 = JMAX(I)
      IF ( J1 .LT. 0 ) GO TO 1610
      DO 1605 J = J1, J2
      INDEX = INDEX + 1
      FREQA( INDEX) = FAL( I,J )
 1605 CONTINUE
 1610 CONTINUE
C
      SEED=IX0
      RETURN
9142  SEED=-1.
      RETURN
      END
C
      SUBROUTINE FGEN (P,NAGE,IAGE,M,S, IL,NFREQ ,IX0, IX1,L1,L2
     *   , L, LINT,IARR )
*****************************************************************************
C
C  THIS SUBROUTINE GENERATES A FISH OF A CERTAIN AGE AND LENGTH
C
*****************************************************************************
      REAL*8 IX0,IX1
      REAL L1, L2
      REAL M(1)
      DIMENSION IARR(20)
      REAL L, LINT
      DIMENSION S(1), P(1)
C       FIRST ASSIGN THE FISH TO AN AGE CLASS.  ******************
        PSUM = 0.
        X=GGUBFS(IX0)
C       CALL RANDU(IX0,IX1,X)
C       IX0 = IX1
```

```
        DO 1100 J= 1,NAGE
          PSUM= PSUM+ P(J)
       IF ( X .LE. PSUM ) GO TO 1200
 1100     CONTINUE
C         THEN GENERATE A NORMAL RANDOM FISH LENGTH   ***************
 1200  IAGE = J
C         CALL RANDU(IX0,IX1,X)
          X=GGUBFS(IX0)
C        IX0=IX1
       IF ( IARR(4) .EQ. 1 ) GO TO 25
C        CALL RANDU(IX0,IX1,Y)
         Y=GGUBFS(IX0)
C         IX0=IX1
       RN= SQRT(-2.*ALOG(X))*COS(2.*3.1415926*Y)
       GO TO 20
    25 CONTINUE
       RN = FNN(X)
    20 CONTINUE
         L=M(J)+RN*S(J)
C         FINALLY PUT THE OBSERVED LENGTH INTO CORRECT FREQUENCY SLOT
         IF ((L .LT. L1 ) .OR. ( L .GT. L2 )) GO TO 1000
           IL = INT(( L-L1)/LINT)+1
         IF ( L .EQ. L2 ) IL = NFREQ
       RETURN
 1000  IL=-1
       RETURN
       END
C
       SUBROUTINE  GETMS(M,S,VBL,VBK,NAGE,MINAG,A,B,VBT,IARR,IX0,FMDEV)
*****************************************************************************
C
C  THIS SUBROUTINE SETS UP THE MEANS AND VARIANCES
C
*****************************************************************************
C
       DIMENSION IARR(20)
       REAL M(1),FMDEV(1)
       REAL*8 IX0
       DIMENSION S(1)
C
       DO 100 I=1,NAGE
         FI= FLOAT(I)
         M(I)= VBL*(1.-EXP(-VBK*(FI-VBT+MINAG-1.)))
         S(I)= A+B*SQRT(FLOAT(I))
   100 CONTINUE
       IF (IARR(15).NE.0)THEN
          SDTEMP=FLOAT(IARR(15))/10.
          DO 43222 I=1,NAGE
            X=GGUBFS(IX0)
            Y=GGUBFS(IX0)
            FMDEV(I)= SQRT(-2.*ALOG(X))*COS(2.*3.1415926*Y)*SDTEMP
            M(I)=M(I)+FMDEV(I)
```

```
43222    CONTINUE
       END IF
       RETURN
      END
C
      FUNCTION FNN(Q)
      IF ( Q .EQ. 0. ) Q= .00001
      IF ( Q .LE. .5 ) T = SQRT( -2. * ALOG(Q))
      IF ( Q .GT. .5 ) T = SQRT( -2. * ALOG( 1.-Q ) )
      Y = T - ( 2.515517 + T * ( .802853 + T * .010328 ) ) /
     *   ( 1.+ T * ( 1.432788 + T * ( .189269 + T *( .001308 ))) )
      X = -Y
      IF ( Q .GT. .5) X = Y
      FNN = X
      RETURN
      END
C
      SUBROUTINE LPOUT(PASIM)
      REAL*8 SEED
      DIMENSION PASIM(1)
      REAL  M(30),L1,L2,L,LINT
            COMMON /OVBLK/          NFREQ,    MINAG,    L1,       LINT,
     *   F(200),  INDEX,    JMIN(100), JMAX(100),FREQA(120),RF(100),
     .   M,         S(30),    P(30),    PSIM(30),  L,       MAXAG,
     .   FAL(100,10),NAGE,   PSUM,     RTOT,     SEED,
     *   SFAL(100,10),INDEXS,SFREQA(25),JSMIN(100),JSMAX(100),
     *   IARR(20),   ZMORT,   RSIG,   FMDEV(20),ITITLE(40),FMORT,
     *   ISEEDX ,  VBK
C
      WRITE (20,2010)
 2010 FORMAT(//,'AGE',T12,'MEAN',T25,'STD DEV', T40,
     *'TRUE %',T60 ,'SIMULATED %')
      MAXAG = MINAG + NAGE - 1
      DO 2015 I=1,NAGE
      II = I + MINAG - 1
      P(I) = 100.* P(I)
      PSIM(I)= 110. * PSIM(I)
      WRITE(20,2020) II,M(I) ,S(I), P(I), PSIM(I)
 2015 CONTINUE
 2020 FORMAT( T3,I3,T12,F8.2,T25,F8.2,T40,F8.2,
     *       T60,F8.2)
      WRITE (20,2030)
 2030 FORMAT(//,T 4 , 'LENGTH', T 25 , 'FREQUENCY' )
      DO 2100 I=1,NFREQ
        L=L1+(FLOAT(I)-.5)*LINT
        WRITE(20,2040)L,F(I)
 2100 CONTINUE
 2040 FORMAT(' ',F15.2,T 25 ,F10.1)
C
C  WRITE FREQUENCIES FOR INPUT TO ANALYSER
      WRITE (21,*) NFREQ, MINAG, L1, LINT
      WRITE (21,2112) (F(I),I=1,NFREQ)
2112  FORMAT (10F7.1)
```

```
C
      RETURN
      END
C
      SUBROUTINE GETPS(NAGE,IXO,ZMORT,RSIG,P)
C
      REAL*8 IXO
      DIMENSION RN(25),P(1)
C
      DO 1000 J=1,NAGE
         X=GGUBFS(IXO)
         Y=GGUBFS(IXO)
         RN(J)=SQRT(-2.*ALOG(X))*COS(2.*3.1415926*Y)
 1000 CONTINUE
C
      POPSUM=0.
      DO 2000 J=1,NAGE
         POPSUM=POPSUM+EXP(RSIG*RN(J)-FLOAT(J)*ZMORT)
 2000 CONTINUE
C
      DO 3000 J=1,NAGE
         P(J)=EXP(RSIG*RN(J)-FLOAT(J)*ZMORT)/POPSUM
 3000 CONTINUE
C
      RETURN
      END
```

POOR

QUALITY

ORIGINAL